

ATARI  
APPLE  
TRS-80 COLOR  
TRS-80 MOD III  
COMMODORE VIC-20  
IBM

---

# THE PERSONAL COMPUTER BASIC[S] REFERENCE MANUAL

---

**Donald A. Sordillo**

# **THE PERSONAL COMPUTER BASIC(S) REFERENCE MANUAL**

**DONALD A. SORDILLO**

**PRENTICE-HALL, INC., Englewood Cliffs, New Jersey 07632**

*Library of Congress Cataloging in Publication Data*

Sordillo, Donald A., (date)

The personal computer BASIC(S) reference manual

1 Microcomputers—Programming. 2 Basic (Computer  
program language) I Title II Title: Personal  
computer B A S I C (S) reference manual

QA76.6 S648 1983 001 64'2 83-9463

ISBN 0-13-658047-5

Editorial/production supervision

and interior design: LYNN FRANKEL

Cover design: PHOTO PLUS ART (Celine A. Brandes)

Manufacturing buyer: GORDON OSBOURNE

© 1983 by Prentice-Hall, Inc., Englewood Cliffs, New Jersey 07632

All rights reserved. No part of this book  
may be reproduced, in any form or by any means,  
without permission in writing from the publisher.

---

NOTE: As of April 1983, IBM released Version 2.0 of BASIC. Since BASIC 1.0 and 1.1 are subsets of Version 2.0, a program written following this book *will* work under the new system. However, a program written using BASIC 2.0 may contain features not described in this book.

The Apple IIe and Franklin 1000 use the same BASICs as Apple Integer and Applesoft. The TRS 64K Color and TRS Color 2 use the same BASICs as TRS Color, Standard and Extended.

---

Printed in the United States of America

10 9 8 7 6 5 4 3 2

ISBN 0-13-658047-5

Prentice-Hall International, Inc., *London*

Prentice-Hall of Australia Pty. Limited, *Sydney*

Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*

Prentice-Hall Canada Inc., *Toronto*

Prentice-Hall of India Private Limited, *New Delhi*

Prentice-Hall of Japan, Inc., *Tokyo*

Prentice-Hall of Southeast Asia Pte. Ltd., *Singapore*

Whitehall Books Limited, *Wellington, New Zealand*

*To Ben Gurley*

ἐχεῖνος ἦν ὁ λύχνος ὁ χαϊόμενος χαὶ φαίνων



# PREFACE

This book is a reference manual of Personal Computer BASIC that covers in depth the IBM, Apple, Atari, TRS Color, TRS Mod III, and Commodore VIC-20 computers. It is intended for the person who:

- Is considering purchasing a personal computer and wants a fast and inexpensive way of comparing functionality among the various systems; or who wants to make sure that a system on the short list is indeed suitable for the intended application.
- Is writing software for commercial purposes and wants to have the product appeal to the widest possible market segment, and not risk cutting out a major part of the market because of an unfortunate choice of instructions when other ones, more widely applicable, are available.
- Wants to convert a program from one system to another and needs a supplement to the computer manufacturer's documentation, especially for the unfamiliar system.
- Like the author, has a mind like a sieve and needs to refer constantly to a reference manual for the rules for the machine on which he or she is currently working.

However, this book does not try to be all things to all people. In particular, it does not attempt to document obscure, obsolete, "big" or foreign machines just so it can boast of being the only volume to document an Urdu dialect of BASIC that is used only by a small band of holy men in the Himalayas. To follow this route would either produce so much useless clutter as to obscure the useful information, or would result in only similarities being described, not differences.

This book has taken the opposite of the "universal" approach. By concentrating only on personal computers and by limiting the number of systems it covers, it is able to document each system fully. The computers covered account for over 90% of personal computers, and include machines with a wide range of functionality (and price), varying from the IBM Personal Computer to "game" machines. In addition, it ties specific functionalities to specific manufacturers—one never has to wonder: "Does *my* machine do this or not?"

One of the aims of the book is to have all the information about a topic in one place. This means *all* the information for *all* the systems covered. There is no "For machine A's peculiarities, see Appendix 1; for machine B's, see Appendix 2; for machine C's, see the manufacturer's documentation; for machine D's, pray." This causes some duplication of material, to be sure; but it relieves the reader from having to flip all over the book to get the necessary information.

By "all the information" is meant just that. All instructions, even input/output instructions and those of the various Disk Operating Systems, are included. (After all, how many programs only compute?) The documentation has been as thorough and comprehensive as possible. Nothing has been intentionally left out because "the field has not yet stabilized." This book identifies which BASIC does *what*. It treats not just the similarities (which are fairly self-evident) but also the differences, especially the subtle ones that make some BASIC systems more equal than others.

Another useful feature is extensive cross-referencing—there are more than 600 cross-references that direct the reader from one topic to others whose existence (or relevance) may not be obvious. Furthermore, there are over 200 charts following the BASIC keywords. These charts not only indicate the format of the command and that each system uses and list which application notes apply, but when different systems use alternate commands to perform the same function, this too is documented.

Only the BASIC language and the statements, commands, and so on, that can be issued from a BASIC program are documented. Command-level aspects of the various operating systems, such as editing and the like, are, in general, not treated. Where meaningful, examples of the instructions have been included. All the examples have been tested; but this does not necessarily mean that the reader's results will be exactly the same.

This book is based on the most up-to-date information, but *omnia mutantur*, especially in the personal computer field. There is no way of predicting what changes a manufacturer will introduce. The final authority has to be the manufacturer's documentation.

As mentioned, this book will be useful to a person who is deliberating over the choice of a system. But a BASIC should not be judged simply by the number of instructions it supports. A better criterion is how easily it can perform the application for which you are buying it. If one's primary purpose is to play games or draw pictures, the most expensive system is not necessarily the best. That is why a perusal of the various charts will prove very useful before one makes a final decision as to which computer to purchase. ANSI minimal BASIC has been included because it is the only industry-wide standard in existence. As one will observe, it is just that: "minimal." There is virtually no system that does not exceed it.

I wish to thank the staff of Computerland in Nashua, New Hampshire, for allowing me "hands on" browsing on many different machines. Special thanks go to the manager, Johnathan Wood, and to Susan Maras. Without their kindness it would have taken much more time (and expense) to produce this book.

The BASIC systems covered by this book, the manufacturers, and the registered trademarks that are referred to are:

**ATARI 400/800**

Atari Corp.  
Sunnyvale, CA 94086  
*Registered trademarks:* ATARI, 400/800.

**Apple, Applesoft, Apple DOS**

Apple Computer, Inc.  
Cupertino, CA 95014  
*Registered trademarks:* Apple, Apple II, Applesoft, Applesoft II.

**TRS Color, TRS Mod III**

Radio Shack  
Fort Worth, TX 76102  
*Registered trademarks:* TRS-80.

**Commodore VIC-20**

Commodore Business Machines, Inc.  
King of Prussia, PA 19406  
*Registered trademarks:* VIC-20.

**IBM**

International Business Machines Corp.  
Boca Raton, FL 33432  
*Registered trademarks:* IBM, MS-DOS.

**Microsoft Consumer Products**

400 108th Ave. NE, Suite 200  
Bellevue, WA 98004

**TO THE READER**

In writing this book certain conventions have been followed; some standard, some invented. These conventions, and tacit assumptions are listed below. They should be read before one tries to use this book, in order to realize its utility fully.

1. Now that you've read this far, before doing anything with the book, read the entry "Format." It will explain how to interpret the formats for the various statements. Since there is no standard system of designating formats, I have tried to use the best from several systems.
2. Entries are in ASCII order. In particular this means that INPUT comes before INPUT #, and TRAP before *trap*.
3. The book was not written as an iron-clad legal document. I have chosen to be colloquial and clear, rather than pedantically precise and obscure. So if a parameter is described as "must be between 0 and 15," a value over 15 or under 0 may cause an error; may not cause an error but may give erroneous results; or may, if positive, be taken modulo 16. The point is that there is nothing to be gained by bloating the book to cover every possible misuse of a statement. For those whose only delight is in showing how, under a highly improbable set of conditions, a certain statement is incorrect, I say, "Enjoy!"
4. The various disk operating systems (DOS) were assumed to include the most powerful BASIC available with the system, even if, strictly speaking, this is not true. That is why, for example, Apple DOS is documented as if all of Applesoft BASIC were part of it, even though it could be used with Integer BASIC.
5. The examples in this book are intended to be just that—illustrations of the various features of BASIC, not a manifestation of programming virtuosity. As such, they are often banal and many would not be used, as is, in a program. On the other hand, by eliminating as much irrelevant matter as possible, they fulfill their purpose—that of demonstrating how a particular feature works. Due to variances among the different systems, the results of numeric examples will probably not be exactly as shown in the book; they should be close, however.
6. Unless otherwise noted, numeric values can be expressions.
7. The term "Carriage Return" has been used to describe the key used to enter values, terminate lines, and the like, even though most terminals do not have a moving carriage. It is the key that generates an ASCII 13; some systems may call it by a different name.
8. The word "CONTROL-" followed by a key means to press the control key and the designated key together. (On some systems the control key may have a different name.) If two or more keys must be pressed together, a plus sign (+) is used between them: CTL + ALT + Z.
9. When characters are associated with an ASCII code, this association applies to screens only. Most modern printers support multiple character sets, so a given code can produce different characters, even on the same printer. Because of this, and the fact that there are a plethora of printers available for personal computers, no attempt has been made to document these codes.
10. Although every effort has been made to make this book accurate, the author and publisher cannot be responsible for changes made by the manufacturers in their quest to improve their products.

## Structure of an Entry

An entry begins with a keyword in the margin, which identifies the entry. If the entry does not concern a BASIC statement, it is usually either a definition or a table or chart. This type of entry is largely self-explanatory. The following discussion deals with those entries that do explain BASIC statements.

Following the keyword is a brief description of what the statement does.

Next is a format which shows how the statement is used in a program. If the statement can appear in more than one discrete way, multiple formats are used. (This is important for statements that are used in two totally different ways, such as GET or PUT.)

Then, a detailed description of how each format operates is given, with programming examples when meaningful.

Following the entry proper is a chart that also contains the keyword. This chart indicates which systems support the statement and, if there are multiple formats for the statement, which format each system supports. If a system does not support the feature, but has the same functionality under a different name, this is also indicated. (If all systems support a feature, or if only one does, then a statement to this effect is put at the end of the description instead of a chart.)

If a system deviates from the general explanation, numerals in the chart indicate which of the notes following the chart are applicable. It is these notes that document the individual differences of each system.

Thus the reader can first get a general idea of the way a statement works, and then find out exactly how the statement works for the particular system or systems of interest.

Following is an example of the charts that follow the entries and an explanation of the various fields.

When there is more than one format, those supported by the system are indicated by number.

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft				
IBM	Cassette				
	Disk				
	Advanced				
TRS Mod III	Level I				
	Exter				

NAME OF ENTRY

An 'X' means the system supports the feature.

When there are differences from the topic as described, they are listed after the chart as a series of notes.

When the same function is called by a different name in another system, it is noted here.

**ABS***Format*

ABS (arithmetic-expression)

The absolute value function, ABS, has as its value the absolute value of the expression.

The absolute value of a number,  $X$  (written  $|X|$ ), is defined as  $X$  if  $X$  is greater than or equal to zero, and as  $-X$  if  $X$  is less than zero. Simply stated, it is the number without its sign. So  $\text{ABS}(X) = \text{ABS}(-X) = X$ .

*Example*

```
100 X = -10
120 Y = ABS(X)
140 PRINT Y, ABS(-15), ABS(X + 15)
```

**Output**

```
10      15      5
```

All BASICs support the ABS function.

*See: function.*

**address**

An address is a fixed location in memory. It can be expressed as *absolute*, in which case it is a numerical value that designates a particular location, or *relative*, in which case it is specified as a positive or negative offset from some fixed reference point.

*Example*

Address	Contents
0100	D
0101	A
0102	S
0103	19

Location 0100 is the *absolute* address of the byte that contains the letter "D." If we use this as our base address or reference point, the *relative*

address of the cell containing the value 19 is +3; that of the one containing the letter “S” is +2. If the location of the letter “S,” 0102, is chosen as the reference point, the relative address of the letter “D” is -2, that of 19 is +1.

## ADR

*Format*

ADR (string-variable)

The address function, ADR, has as its value the address, in decimal, of the specified string.

The ADR function is supported only by ATARI.

*See: VARPTR, VARPTR\$.*

## algorithm

An algorithm is a set of instructions which, if followed, will guarantee an answer to any problem of a given type. For example, given a number,  $X$ , it is desired to find the value of  $e^X$ , where  $e$  is the base of the natural logarithms. The following algorithm solves the problem for any value of  $X$ . This algorithm makes use of the fact that

$$e^X = 1 + X + \frac{X^2}{2!} + \frac{X^3}{3!} + \frac{X^n}{n!} \cdots$$

- Step 1.** Set  $I = 0$  and  $SUM = 0$ .
- Step 2.** Calculate  $X^I/I!$  (Recall that  $0! = 1$ .)
- Step 3.** Add the result of step 2 to  $SUM$ .
- Step 4.** If the value of  $SUM$  is at the desired precision, stop. If not, continue with step 5.
- Step 5.** Add 1 to  $I$ .
- Step 6.** Go back to step 2.

An algorithm for playing a single game of dice is:

- Step 1.** Roll the dice and read the sum.
- Step 2.** If the sum is 2, 3, or 12, you lose.
- Step 3.** If the sum is 7 or 11, you win.
- Step 4.** If the sum is anything else, it is your “point”; continue below.
- Step 5.** Roll the dice and read the sum.
- Step 6.** If the sum is 7, you lose.
- Step 7.** If the sum equals the point, you win.
- Step 8.** If you have neither won nor lost, go to step 5.

**alphabetic character**

An alphabetic character is any of the characters

{A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q,  
R, S, T, U, V, W, X, Y, Z}

In some implementations the lowercase letters are also considered alphabetic characters.

*See: numeric character.*

**alpha-numeric character**

An alphanumeric character is any of the characters

{A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q,  
R, S, T, U, V, W, X, Y, Z, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

In some implementations the lowercase letters are also considered alphanumeric characters. In IBM documentation this class is sometimes called "alphameric."

*See: numeric character.*

**AND**

*Format*

argument-1 AND argument-2

AND is a logical function of two arguments. It has a value of true if both of the arguments are true and a value of false, otherwise.

The arguments can be relations, logical variables, or anything that can be evaluated as true or false. For example:

(X > 5) AND (Y < YMAX)  
P AND Q

Truth Table for AND

p	q	p AND q
F	F	F
F	T	F
T	F	F
T	T	T

*Example*

IF X > 5 AND Y < = YMAX THEN GOTO 100

In this example, the GOTO is executed only if, when the relation is evaluated, both X is greater than 5, and Y is less than or equal to YMAX.

All BASIC systems support AND.

*See: logical functions, NOT, OR.*

## APPEND

*Format*

APPEND file-name [,Ddrive] [,Sslot] [,Vvolume]

The APPEND command opens the specified file and positions the record pointer to the end of the file. Data subsequently written to the file will be put just after the last data currently in the file. If a READ is executed after an APPEND, it will cause an error. APPEND should not be followed by an OPEN statement since OPEN sets the record pointer to the beginning of the file.

This command must be executed, preceded by a CONTROL-D, as part of a PRINT statement

PRINT CHR\$(4); "APPEND MYFILE,D1,S6"

The allowable values for parameters and their defaults are shown below. Once a value has been changed, the last value specified is the default.

Parameter	Range	Default
drive	1 or 2	Drive 1
slot	1 to 7	Slot from which DOS was booted
volume	1 to 254	Volume from which DOS was booted

APPEND is supported only by Apple DOS.

*See: OPEN.*

## argument

An argument is the independent variable of a function: that is, the value that is supplied to the function. The function, in turn, manipulates the argument to obtain a result which is the function's value for that argument.

For example, consider the square root function in the statement  $Y = \text{SQR}(X)$ . The function is SQR, its argument is X; its value, which will be assigned to Y, depends on the value chosen for X.

A function can have zero, one, or multiple arguments.

*See: dummy variable.*

**arithmetic expression**

An arithmetic expression is:

1. A numeric constant, or
2. A numeric variable, or
3. A numeric function, or
4. Two numeric constants, variables, and/or functions combined by an arithmetic operator, or
5. An arithmetic expression in parentheses combined with any other arithmetic expression.

Parentheses may be placed around an arithmetic expression without changing its value. They may, however, change the way the expression is evaluated. For example,  $A * B + C$  is evaluated as  $(A * B) + C$ . But if parentheses are placed thus:  $A * (B + C)$ , the sum  $B + C$  is evaluated first and the result is multiplied by  $A$ .

*Examples of Arithmetic Expressions*

5  
X  
ABS(X + 5)  
X + 5  
 $(X + 5) * (Y/Z)$

*See: arithmetic operations, arithmetic operator.*

**arithmetic operations**

The arithmetic operations of addition, subtraction, multiplication, division, and exponentiation are supported by all BASICs. In addition, the operations of modulus and integer division are supported by some implementations.

Relations between arithmetic expressions and the symbols used to represent them are:

Equality	=
Greater than	>
Less than	<
Greater than or equal to	> = or = >
Less than or equal to	< = or = <
Not equal	> < or < >

The evaluation of an arithmetic expression depends on the precedence of the individual operations. The following list summarizes the precedence rules and other features of the various systems.

*Precedence Rules***Set 1**

Innermost parentheses  
Exponentiation  
Unary plus and minus  
Multiplication and division  
Addition and subtraction  
Relational operators  
Logical operators: NOT, AND, OR

**Set 2**

Innermost parentheses  
Arithmetic functions  
Relational functions  
Logical functions  
Exponentiation  
Unary plus and minus  
Multiplication and division  
Integer division  
MOD  
Addition and subtraction  
Relational operators  
Logical operators: NOT, AND, OR, XOR, IMP, EQV

**Set 3**

Innermost parentheses  
Unary plus and minus, and NOT  
Exponentiation  
Multiplication, division, and MOD  
Addition and subtraction  
Relational operators  
Logical operators: AND, OR

**Set 4**

Relation operators in *string* expressions  
Unary plus and minus  
Exponentiation  
Multiplication and division

Addition and subtraction

Relation operators in *numeric* expressions

NOT

AND

OR

The accompanying table shows which set of precedence rules an implementation follows, how values are converted to integers, the precision in which mixed-mode operations are carried out, and the symbol used for exponentiation.

*See: logical functions.*

arithmetic operations

System		Notes	Precedence Rules	Converts to Integer by:	Precision of Mixed mode	Exponential Symbol
APPLE	Integer	1	3	Truncation	Not applicable	^
	Applesoft	2	3	Rounding or truncation	Single	^
	DOS		3	Rounding or truncation	Single	^
	Microsoft		2	Rounding	Most precise operand	↑
IBM	Cassette		2	Rounding	Most precise operand	^
	Disk		2	Rounding	Most precise operand	^
	Advanced		2	Rounding	Most precise operand	^
TRS Mod III	Level I		1	Truncation	Most precise operand	[ or ↑
	Extended	3	1	Truncation	Most precise operand	[ or ↑
	Disk	3	1	Truncation	Most precise operand	[ or ↑
TRS Color	Level I		1	Truncation	Single	↑
	Extended		1	Truncation	Single	↑
	Disk		1	Truncation	Single	↑
Commodore	VIC 20		1	Truncation	Single	↑
ATARI	400/800		4	Rounding	Not applicable	^
ANSI	Minimum		1			

### Notes

1. Uses the number sign (#) for "not equal."
2. MOD is not supported.
3. Division is single or double precision; exponentiation is single precision.

**arithmetic operator**

An arithmetic operator is an operator that takes an arithmetic expression as an operand. The arithmetic operators and the operations they represent are:

Addition	+
Subtraction	—
Multiplication	*
Division	/
Exponentiation	^ or ↑
Modulo	MOD
Integer division	\

*See: arithmetic expression, arithmetic operations.*

**array**

An array is a set of logically consecutive data items with a common name. Individual elements of an array are referenced by a value termed a “subscript.”

The size of an array is specified in the DIM statement. The statement DIM A(20) defines an array named A, which has 21 elements designated 0 to 20. To access a specific element of an array, its position in the array is indicated by a subscript, which is an arithmetic expression written after the name of the array, in parentheses. The OPTION BASE statement specifies whether the lowest accessible element is the zeroth or first. If an array is not dimensioned, the default is 11 elements, numbered 0 to 10.

*Examples*

The statement A(5) = 32 sets the fifth element of array A to the value 32.

If X = 3, the statement A(X \* 2) = Y sets the sixth element of array A to the value of Y.

If an implementation allows it, an array can have more than one dimension. An *n*-dimensional array is specified by

DIM array-name (value-1,value-2,value-3, ... ,value-*n*)

where value-1 specifies the size of the first dimension, value-2, the size of the second dimension, and so on.

The accompanying table summarizes some properties of arrays. The “Range of Elements” is for an array specified as DIM A(N). Where values for dimensions and elements per dimension are not given, the limits depend to a large degree on the available memory and a hard and fast rule cannot be given.

*See: DIM, ERASE, OPTION BASE, subscript.*

array

System			Notes	Range of Elements	Maximum Elements per Dimension	Maximum Dimensions
APPLE	Integer		1, 2	1 to N		1
	Applesoft			0 to N		88
	DOS			0 to N		88
	Microsoft			0 or 1 to N	32767	255
IBM	Cassette			0 or 1 to N	32767	255
	Disk			0 or 1 to N	32767	255
	Advanced			0 or 1 to N	32767	255
TRS Mod III	Level I			0 to N		
	Extended			0 to N		
	Disk			0 to N		
TRS Color	Level I			0 to N - 1		
	Extended			0 to N - 1		
	Disk			0 to N - 1		
Commodore	VIC 20			0 to N		
ATARI	400/800		1	0 to N	32767	2
ANSI	Minimum			0 or 1 to 10		

*Notes*

1. Does not support string arrays; DIM A\$(10) defines the variable A\$ as having a size of 10 characters.
2. A(0) is considered identical to A; arrays must be cleared by the program.

**ASC***Format*

ASC (string-variable)

The ASC function returns as its value the ASCII code of the first character of the specified string. The value returned is in decimal. If the string-variable is null, it causes an error. ASC is the inverse function of CHR\$.

*Example*

```

100 X$ = "ABC"
120 Y = ASC(X$)
140 PRINT Y, ASC("Z")

```

**Output**

65     90   (65 is the code for “A”; 90 for “Z”)

*See: ASCII codes, CHR\$.*

ASC
-----

System		In	Format	Notes	Alternate Commands
APPLE	Integer	X		1	
	Applesoft	X		2	
	DOS	X		2	
	Microsoft	X			
IBM	Cassette	X			
	Disk	X			
	Advanced	X			
TRS Mod III	Level I	X			
	Extended	X			
	Disk	X			
TRS Color	Level I	X			
	Extended	X			
	Disk	X			
Commodore	VIC 20	X			
ATARI	400/800	X			
ANSI	Minimum				

*Notes*

1. If the value is over 127, subtract 128 from it to get the true character.
2. A CONTROL-@ cannot be used as the argument.

**ASCII codes**

ASCII is an acronym for American Standard Code for Information Interchange. The code defines 256 symbols and functions. Different manufacturers may choose to assign to a particular code a symbol or function unique to its implementation.

In particular, printers generally have their own built-in tables for generating a character from an ASCII code. These characters may be different from the ones that would be displayed on the screen for the same code.

Following is a table of the codes that are common to all BASICs and the associated characters. These codes apply to the screen and keyboard, but not necessarily to printers.

Code (Dec)	Code (Hex)	Character
32	20	space
33	21	!
34	22	"
35	23	#
36	24	\$
37	25	%
38	26	&
39	27	apostrophe
40	28	(
41	29	)
42	2A	*
43	2B	+
44	2C	comma
45	2D	hyphen
46	2E	period
47	2F	/
48	30	0
49	31	1
50	32	2
51	33	3
52	34	4
53	35	5
54	36	6
55	37	7
56	38	8
57	39	9
58	3A	colon
59	3B	semicolon
60	3C	<
61	3D	=
62	3E	>
63	3F	?
64	40	@

Code (Dec)	Code (Hex)	Character
65	41	A
66	42	B
67	43	C
68	44	D
69	45	E
70	46	F
71	47	G
72	48	H
73	49	I
74	4A	J
75	4B	K
76	4C	L
77	4D	M
78	4E	N
79	4F	O
80	50	P
81	51	Q
82	52	R
83	53	S
84	54	T
85	55	U
86	56	V
87	57	W
88	58	X
89	59	Y
90	5A	Z
91	5B	[
92	5C	\
93	5D	]
94	5E	
95	5F	underline
96	60	
97	61	a

Code (Dec)	Code (Hex)	Character
98	62	b
99	63	c
100	64	d
101	65	e
102	66	f
103	67	g
104	68	h
105	69	i
106	6A	j
107	6B	k
108	6C	l
109	6D	m
110	6E	n
111	6F	o
112	70	p
113	71	q
114	72	r
115	73	s
116	74	t
117	75	u
118	76	v
119	77	w
120	78	x
121	79	y
122	7A	z

*Notes***TRS Mod III**

Codes 0 to 31 are graphics symbols.

Codes 97 to 122 (lowercase letters) can be generated by using SHIFT + the uppercase letter.

Codes 128 to 255 are graphics symbols.

**Apple II**

Codes 96 to 127 generate the same characters as codes 32 to 63.

Codes 128 to 223 generate the same characters as codes 0 to 95.

Codes 224 to 255 generate the same characters as codes 0 to 31.

**TRS Color**

Codes 97 to 122 generate the same characters as codes 65 to 90, but the color is reversed. They are generated by pressing SHIFT + 0 + one of the letters A to Z. Codes 128 to 255 are graphics characters.

**VIC 20: Text Mode**

Codes 192 to 223 generate the same characters as codes 96 to 127.

Codes 224 to 254 generate the same characters as codes 160 to 190.

Code 255 generates the same character as code 126.

For the VIC-20, 92 is “£,” 94 is “↑,” and 95 is “←.”

**ATARI**

Codes 0 to 31 and 96 are graphics symbols.

**assembly language**

An assembly language is a symbolic language that, in general, has the property that one instruction in assembly language is translated into one machine instruction. (In a language such as BASIC, one statement is usually equivalent to several machine instructions.)

Some assemblers have a *macro* capability; that is, one instruction can represent several machine instructions. When the program is assembled, the macro is *expanded* into its equivalent instructions, but the resultant program still results in a one-to-one relationship between assembly and machine instructions. Thus the effect of a macro is to reduce the number of lines the programmer has to write, not to reduce the size of the object code. Assembly language routines can be accessed from a BASIC program by CALL and USR.

*See: CALL, USR.*

**assignment statement**    *See: LET.*

**ATN**                      *Format*

ATN (arithmetic-expression)

The arctangent function, ATN, has as its value an angle expressed in radians, whose tangent is equal to the value of the expression. The angle returned is in the range  $-\pi/2$  to  $\pi/2$  radians ( $-90$  to  $+90$  degrees). This is the inverse of the tangent function. To get degrees, multiply the result by 57.29578.

*Example*

```
100 XR = .5773503
120 Y = ATN(XR)
140 PRINT Y, ATN(XR) * 57.29578
```

**Output**

.5235988 (radian)      30 (degrees)

*See: COS, DEG, RAD, SIN, TAN, trigonometric functions.*

ATN

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft	X			
	DOS	X			
	Microsoft	X			
IBM	Cassette	X			
	Disk	X			
	Advanced	X			
TRS Mod III	Level I				
	Extended	X			
	Disk	X			
TRS Color	Level I				
	Extended	X			
	Disk	X			
Commodore	VIC 20	X			
ATARI	400/800	X		1	
ANSI	Minimum	X			

*Note*

1. The argument can be in degrees or radians, depending on whether DEG or RAD was executed.

**AUDIO***Format*

AUDIO OFF|ON

The AUDIO statement connects or disconnects the output of the cassette to the TV speaker.

AUDIO is supported only by TRS Color (all levels).

*See: BEEP, PLAY, SOUND.*

**AUTO***Format*

AUTO [line-number] [,increment]

The AUTO command allows automatic line numbering when the program is being typed in. The first program line is assigned the specified line-number, and each subsequent line is assigned the preceding line-number plus the increment. The default is 10,10.

*Example*

The statement

AUTO 100,20

will result in the program lines being numbered 100, 120, 140, and so on.

*See: RENUM.*

AUTO

System		In	Format	Notes	Alternate Commands
APPLE	Integer	X		5	
	Applesoft				
	DOS				
	Microsoft	X		1, 3	
IBM	Cassette	X		1, 4	
	Disk	X		1, 4	
	Advanced	X		1, 4	
TRS Mod III	Level I				
	Extended	X		2	
	Disk	X		2	
TRS Color	Level I				
	Extended				
	Disk				
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

### Notes

1. If a line already has something on it, an asterisk (\*) is printed next to it. To leave it alone, type a Carriage Return.
2. If a line already has something on it, an asterisk (\*) is printed next to it. To leave it alone, press the BREAK key.
3. Use CONTROL-C to leave AUTO mode.
4. Use CONTROL-BREAK to leave AUTO mode.
5. Press CONTROL-X and then type MAN to leave AUTO mode.

# B

## **BACKUP**      *Format*

BACKUP source TO destination

The BACKUP command copies the disk on the source drive onto the disk on the destination drive. If the system has only one drive, it should be specified as the source, and the system will issue the necessary prompts. Executing this command erases memory.

“Source” and “destination” can have values of 0 to 3.

BACKUP is supported only by TRS Color DOS.

**BASIC**      BASIC is an acronym for Beginner’s All-purpose Symbolic Instruction Code.

**baud**      Baud is a unit used to measure the rate of transfer of information. In most modern applications, baud is equivalent to “bits per second.”

**BEEP**      *Format*

BEEP [pitch, duration]

The BEEP statement generates a tone for a certain amount of time. *Pitch* can have values from 0 to 255, with 255 representing the lowest tone; *duration* can have values from 0 to 255, with 255 representing the longest duration, equal to about 1 second.

*See: AUDIO, PLAY, SOUND.*

BEEP

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft	X			
IBM	Cassette	X		1	
	Disk	X		1	
	Advanced	X		1	
TRS Mod III	Level I				
	Extended				
	Disk				
TRS Color	Level I				SOUND
	Extended				SOUND
	Disk				SOUND
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

*Note*

1. BEEP generates an 800-hertz tone for  $\frac{1}{4}$  second. It is equivalent to PRINT CHR\$(7). The pitch and duration cannot be specified.

**binary digit***See: bit.***binary number**

A binary number is represented by a string of ones and zeros. Each position in the number indicates the presence or absence of a power of 2.

*Example*

The binary number 10110 is evaluated as follows:

$$\begin{array}{cccccc}
 1 & & 0 & & 1 & & 1 & & 0 \\
 (1 \times 2^4) & + & (0 \times 2^3) & + & (1 \times 2^2) & + & (1 \times 2^1) & + & (0 \times 2^0) \\
 16 & + & 0 & + & 4 & + & 2 & + & 0 \\
 & & & & & & & & = 22 \text{ (base 10)}
 \end{array}$$

*See: conversion tables, hexadecimal number, octal number.*

**bit**

The word “bit” is a contraction of “binary digit.” In programming applications, it has come to mean the smallest entity that has meaning. A bit can be in only one of its two possible states at any given time. Depending on the interpretation, these states can be 0 or 1, ON or OFF, or TRUE or FALSE.

*See: byte, nibble.*

**BLOAD**

*Format 1*

BLOAD file-specification [,offset]

*Format 2*

BLOAD file-name [,Address] [,Ddrive] [,Sslot] [,Vvolume]

The BLOAD command loads a machine language program that was previously stored with a BSAVE into memory.

*Format 1*

In Cassette BASIC, if no device is specified, CAS1: (the only valid device) is used. The motor is turned on and a search is made for the specified file. For Disk and Advanced BASIC, if no device is specified, the current disk is used.

The offset is an arithmetic expression with a value between 0 and 65535. It specifies the address at which the program is loaded as an offset into the current segment. If no offset is specified, the program is loaded into the same area it was saved from.

Use of BLOAD is not restricted to programs; any part of memory can be specified by using the DEF SEG statement. In particular, the screen buffer area can be accessed.

*Example*

BLOAD “MYFILE.EXE”, 0

*Format 2*

The program specified by file-name is loaded into memory starting at the specified address. The program is *not* run.

This command must be executed, preceded by a CONTROL-D, as part of a PRINT statement:

PRINT CHR\$(4); "BLOAD MYFILE, A1000, D1, S6, V10"

The allowable values for parameters and their defaults are shown below. Once a value has been changed, the last value specified is the default.

Parameter	Range	Default
address	0 to 65535	Area program BSAVED from
drive	1 or 2	Drive 1
slot	1 to 7	Slot from which DOS was booted
volume	1 to 254	Volume from which DOS was booted

*See: BRUN, BSAVE, CLOAD, CLOADM, DEF SEG, file specification.*

BLOAD
-------

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS	X	2		
	Microsoft				
IBM	Cassette	X	1		
	Disk	X	1		
	Advanced	X	1		
TRS Mod III	Level I				
	Extended				
	Disk				
TRS Color	Level I				
	Extended				
	Disk				
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

## Boolean functions

*See: logical functions.*

## BRUN

*Format*

BRUN file-name [,Aaddress] [,Ddrive] [,Sslot] [,Vvolume]

The BRUN command loads the designated file into memory starting at the

specified address. Once loaded, control is transferred to the starting address of the routine.

This command must be executed, preceded by a CONTROL-D, as part of a PRINT statement:

```
PRINT CHR$(4); "BRUN MYFILE, A1000, D1, S6"
```

The allowable values for parameters and their defaults are shown below. Once a value has been changed, the last value specified is the default.

Parameter	Range	Default
address	0 to 65535	Address program BSAVED from
drive	1 or 2	Drive 1
slot	1 to 7	Slot from which DOS was booted
volume	1 to 254	Volume from which DOS was booted

BRUN is supported only by Apple DOS.

*See: BLOAD, BSAVE, CALL, DEF SEG, file specification, RUN.*

## BSAVE

### *Format 1*

BSAVE file-specification, offset, length

### *Format 2*

BSAVE file-name, Aaddress, Llength  
[ ,Ddrive ] [ ,Sslot ] [ ,Vvolume ]

The BSAVE command stores a machine language program on external media.

### *Format 1*

In Cassette BASIC, if no device is specified, CAS1: (the only valid device) is used. In Disk and Advanced BASIC, if no drive is specified, the current disk is used.

The contents of memory, beginning at the location indicated by the offset and continuing for the specified number of bytes, are saved and given the name designated in the file-specification.

Use of BSAVE is not restricted to programs. Any part of memory can be saved, in particular the screen buffer area.

The offset must have a value of 0 to 65535; it specifies the offset into the segment last declared by the DEF SEG statement. The length must have a value of 1 to 65535. It specifies the number of bytes to store.

### *Example*

```
BSAVE "MYFILE.EXE", 0, &H1000
```

### *Format 2*

The contents of memory beginning at the specified address and for the specified length are stored and given the file-name.

This command must be executed, preceded by a CONTROL-D, as part of a PRINT statement:

```
PRINT CHR$(4); "BSAVE MYFILE, A1000, L512, D1, S6"
```

The allowable values for parameters and their defaults are shown below. Once a value has been changed, the last value specified is the default.

Parameter	Range	Default
address	0 to 65535	None
length	1 to 32767	None
drive	1 or 2	Drive 1
slot	1 to 7	Slot from which DOS was booted
volume	1 to 254	Volume from which DOS was booted

***See: BLOAD, BRUN, CSAVE, CSAVEM, DEF SEG, file specification, SAVE.***

BSAVE

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS	X	2		
	Microsoft				
IBM	Cassette	X	1		
	Disk	X	1		
	Advanced	X	1		
TRS Mod III	Level I				
	Extended				CSAVEM
	Disk				CSAVEM
TRS Color	Level I				
	Extended				CSAVEM
	Disk				CSAVEM
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

**BUTTON***Format*

BUTTON (arithmetic-expression)

The **BUTTON** function returns the current value of the push button on the specified game controller. If the button is not pressed, a zero (FALSE) is returned. If it is pressed, a  $-1$  (TRUE) is returned.

The arithmetic-expression must be an integer between 0 and 3; it specifies which controller to test.

*Example*

```

100 IF BUTTON (0) THEN GOTO 200
120 IF BUTTON (1) THEN GOTO 300
140 GOTO 100

```

This loop waits for a button to be pressed and then branches to a servicing routine.

*See: JOYSTK, PADDLE, PDL, PTRIG, STICK, STRIG.*

BUTTON

System	In	Format	Notes	Alternate Commands
APPLE	Integer			
	Applesoft		1	PEEK
	DOS		1	PEEK
	Microsoft	X		
IBM	Cassette			STRIG
	Disk			STRIG
	Advanced			STRIG
TRS Mod III	Level I			
	Extended			
	Disk			
TRS Color	Level I		2	PEEK
	Extended		2	PEEK
	Disk		2	PEEK
Commodore	VIC 20			
ATARI	400/800			
ANSI	Minimum			

*Notes*

1. PEEK (−16287) is button 0; PEEK (−16286) is button 1; PEEK (−16285) is button 2. If the value returned is over 127, the button is being pressed.
2. PEEK (65280). If a button is not being pressed, a value of 255 or 127 is returned. If the right-hand button is being pressed, a value of 126 or 254 is returned. If the left-hand button is being pressed, a value of 125 or 253 is returned.

**BYE**

*Format*

BYE

The BYE statement exits BASIC and enters the Screen Pad Mode. To return to BASIC, use SYSTEM RESET.

BYE is supported only by ATARI.

*See: END, STOP.*

**byte**

A byte is a group of bits treated as an entity. In most personal computers a byte consists of 8 bits and is the smallest addressable unit of memory.

However, in some specialized communications applications, a byte may contain fewer or more bits.

The size of a storage device (memory, tape, or disk) is generally described in terms of the number of kilobytes (K-bytes, or simply K) it contains. In this context, the term “K” refers to 1024 bytes, as opposed to its usual meaning of 1000.

*See: bit, nibble.*

**CALL***Format 1*

CALL numeric-variable [({argument,} ... )]

*Format 2*

CALL arithmetic-expression

*Format 3*

CALL %variable-name [({argument,} ... )]

The CALL statement is used to invoke a machine language routine and to pass parameters to it.

*Format 1*

The value of the numeric-variable specifies the starting address of the routine being called. If arguments are specified, they are passed to the routine.

*Format 2*

The expression indicates the starting address of the routine; it must be between 0 and 65535. No parameters can be passed.

*Format 3*

This form of CALL is used to invoke 6502 subroutines. A percent sign (%) must precede the numeric variable. Up to three 1-byte parameters can be passed. The first one goes to the A-register; the second to the X-register; and the last to the Y-register.

***See: BRUN, CLOADM, DEF SEG, EXEC, SYS, USR.***

CALL

System		In	Format	Notes	Alternate Commands
APPLE	Integer	X	2	1	
	Applesoft	X	2	2	
	DOS	X	2	2	
	Microsoft	X	1, 3	3	
IBM	Cassette	X	1	4	
	Disk	X	1	4	
	Advanced	X	1	4	
TRS Mod III	Level I				
	Extended				
	Disk				
TRS Color	Level I				EXEC
	Extended				EXEC
	Disk				EXEC
Commodore	VIC 20				SYS
ATARI	400/800				
ANSI	Minimum				

*Notes*

1. Addresses over 32767 must be represented by the address minus 65536.
2. Addresses can be between -65535 and 65535; a negative address is considered equivalent to the corresponding positive one.
3. The variable cannot be an array.
4. The numeric-variable specifies the starting address as an offset into the current memory segment as defined by the last DEF SEG statement.

**CATALOG** *Format*CATALOG [*Ddrive*] [*Sslot*]

The CATALOG command displays the volume number and a list of all the files on the disk in the specified drive. The file type and the number of sectors it occupies are also displayed. (If the file occupies more than 255 sectors, the value is given modulo 256.)

The codes used for the file types are:

Code	Meaning
I	Integer BASIC file, created by SAVE
A	Applesoft BASIC file, created by SAVE
T	Text file created by OPEN, filled by WRITE
B	Binary file, created by BSAVE

(If the file is locked, an asterisk is displayed beside the file type.)

This command must be executed, preceded by a CONTROL-D, as part of a PRINT statement:

```
PRINT CHR$(4); "CATALOG D1, S6"
```

The allowable values for parameters and their defaults are shown below. Once a value has been changed, the last value specified is the default.

Parameter	Range	Default
drive	1 or 2	Drive 1
slot	1 to 7	Slot from which DOS was booted

CATALOG is supported only by Apple DOS.

*See: DIR, FILES.*

## CDBL

*Format*

CDBL (arithmetic-expression)

The convert-to-double-precision function, CDBL, has as its value the double-precision representation of the argument. Although the value is double precision, only the number of digits that existed in the argument are significant.

*Example*

```

100 DEFDBL D
120 DEFINT I
140 S = 12345.67
160 I = 12345
180 D = CDBL(S)
200 PRINT CDBL(I), D

```

*Output*

```

12345      12345.66992185

```

(The actual value of D will differ among implementations.)

*See: CINT, CSNG.*

CDBL
------

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft	X			
IBM	Cassette	X			
	Disk	X			
	Advanced	X			
TRS Mod III	Level I				
	Extended	X			
	Disk	X			
TRS Color	Level I				
	Extended				
	Disk				
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

**CHAIN***Format 1*

```

CHAIN [MERGE] file-specification [,line-number]
      [,ALL] [,DELETE first-line [- last-line]]

```

*Format 2*

```

CHAIN file-name [,Ddrive] [,Sslot] [,Vvolume]

```

The CHAIN command loads and runs the specified BASIC program and, optionally, passes it variables.

#### *Format 1*

The program named in the file specification is loaded. If MERGE is specified (in which case the chained program must have been saved with the "A" option), a merge operation is performed and the program overlays the program currently in memory. All files currently open are left open, the current OPTION BASE setting is maintained, and all variable type and function definitions are preserved (DEFINT, DEFSNG, DEFDBL, DEFSTG, DEF FN). If MERGE is not specified, these variable type and function definitions are not preserved.

The line-number is an arithmetic-expression that designates the line at which execution begins when the chained program is loaded. If no line-number is specified, execution begins at the first line of the program. (A RENUM command does not change this number.)

#### **ALL**

If ALL is specified, every variable in the current program is passed to the chained program. If ALL is not specified, any variable to be passed to the chained program must be specified in a COMMON statement.

#### **DELETE**

If DELETE is specified, the indicated line or lines in the program currently in memory are deleted before the new program is loaded. (These line numbers are affected by RENUM.)

If only first-line is specified, that line is deleted. If the hyphen and last-line are specified, all lines from the start of the program to last-line, inclusive, are deleted. If both first-line and last-line are specified, all lines from the first to the last, inclusive, are deleted.

#### *Example (no parameters)*

```
100 REM THIS IS THE "FIRST" PROGRAM
120 PRINT "PROGRAM 1"
140 CHAIN "SECOND"
160 END
```

```
100 REM THIS IS THE "SECOND" PROGRAM
120 PRINT "PROGRAM 2"
140 CHAIN "FIRST"
160 END
```

**Output after Program 1 Is Run**

```

PROGRAM 1
PROGRAM 2
PROGRAM 1
etc. (must be manually interrupted)

```

*Example (COMMON and parameters)*

```

100 REM THIS IS PROGRAM 1
120 COMMON A, X$, B()
140 DIM B(5)
160 A = 10 : X$ = "ABC" : D = -3.5
180 B(1) = 17.32 : B(5) = 50.5
200 PRINT "PROGRAM 1'S VARIABLES (BEFORE): "
220 PRINT A, X$, B(1), B(5), D
240 PRINT
260 CHAIN "A:PROG2"
280 PRINT "PROGRAM 1'S VARIABLES (AFTER): "
300 PRINT A, X$, B(1), B(5), D
320 END

100 REM THIS IS PROGRAM 2
120 COMMON A, X$, B()
140 PRINT "PROGRAM 2'S VARIABLES ARE: "
160 PRINT A, X$, B(1), B(5), D
180 PRINT
200 CHAIN "B:PROG1", 280
220 END

```

**Output after Loading PROG1 in Memory and Running It**

```

PROGRAM 1'S VARIABLES (BEFORE):
10   ABC    17.32   50.5   -3.5

PROGRAM 2'S VARIABLES ARE:
10   ABC    17.32   50.5     0

PROGRAM 1'S VARIABLES (AFTER):
10   ABC    17.32   50.5     0

```

*Example (COMMON statements that are different)*

```

100 REM THIS IS PROGRAM 1
120 COMMON A, X$, B()

```

```

140 DIM B(5)
160 A = 10 : X$ = "ABC" : D = - 3.5
180 B(1) = 17.32 : B(5) = 50.5
200 PRINT "PROGRAM 1'S VARIABLES (BEFORE): "
220 PRINT A, X$, B(1), B(5), D
240 PRINT
260 CHAIN "A:PROG2"
280 PRINT "PROGRAM 1'S VARIABLES (AFTER): "
300 PRINT A, X$, B(1), B(5), D
320 END

100 REM THIS IS PROGRAM 2
120 COMMON D
140 D = 123.45
160 PRINT "PROGRAM 2'S VARIABLES ARE: "
180 PRINT A, X$, B(1), B(5), D
200 PRINT
220 CHAIN "B:PROG1", 280
240 END

```

### Output after Running Program 1

PROGRAM 1'S VARIABLES (BEFORE):

10	ABC	17.32	50.5	- 3.5
----	-----	-------	------	-------

PROGRAM 2'S VARIABLES ARE:

10	ABC	17.32	50.5	123.45
----	-----	-------	------	--------

PROGRAM 1'S VARIABLES (AFTER):

0	0	0	123.45	(X\$ is null)
---	---	---	--------	---------------

### Format 2

In this form of CHAIN, parameters are not explicitly passed. However, current variables are not cleared, so the chained program can access any of them.

This command must be executed, preceded by a CONTROL-D, as part of a PRINT statement:

```
PRINT CHR$(4); "CHAIN MYFILE, D1, S6, V2"
```

The allowable values for parameters and their defaults are shown below. Once a value has been changed, the last value specified is the default.

Parameter	Range	Default
drive	1 or 2	Drive 1
slot	1 to 7	Slot from which DOS was booted
volume	1 to 254	Volume from which DOS was booted

*See: COMMON, MERGE, RENUM, SAVE.*

CHAIN
-------

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS	X	2		
	Microsoft	X	1		
IBM	Cassette				
	Disk	X	1		
	Advanced	X	1		
TRS Mod III	Level I				
	Extended				
	Disk				
TRS Color	Level I				
	Extended				
	Disk				
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

## CHR\$

*Format*

CHR\$ (arithmetic-expression)

The string function CHR\$ has as its value the single ASCII character corresponding to the value of the expression. It is the inverse function of ASC.

The value of the expression must be between 0 and 255. The actual character returned is dependent on the hardware. (See the entry "ASCII codes.")

Values of 0 to 31 are control characters; if one of these is printed, the function it represents is performed. For example, PRINT CHR\$(7) produces a tone.

*Example*

```
100 X$ = CHR$(65)
120 PRINT X$, CHR$(66)
```

**Output**

A     B

*See: ASC.*

CHR\$
-------

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft	X		1	
	DOS	X		1	
	Microsoft	X			
IBM	Cassette	X			
	Disk	X			
	Advanced	X			
TRS Mod III	Level I				
	Extended	X			
	Disk	X			
TRS Color	Level I	X			
	Extended	X			
	Disk	X			
Commodore	VIC 20	X			
ATARI	400/800	X		2	
ANSI	Minimum				

*Notes*

1. Codes 96 to 255 generate characters that repeat codes 0 to 95; but even though CHR\$(65) and CHR\$(193) both return an "A," they are considered as different when used in relations.
2. A relation can contain at most one CHR\$; that is, CHR\$(X) > = CHR\$(Y) is invalid.

**CINT***Format*

CINT (arithmetic-expression)

The convert to integer function, CINT, returns as its value the integer obtained by rounding the arithmetic-expression. This expression must evaluate within the range -32768 to +32767.

*Example*

```

100 DEFINT I
120 DEFDBL D
140 D = 12345.178901234
160 S = -1234.568
180 I = CINT(S)
200 PRINT CINT(D), I

```

**Output**

```

12345      -1235

```

*See: CDBL, CSNG, FIX, INT.*

CINT

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft	X			
IBM	Cassette	X			
	Disk	X			
	Advanced	X			
TRS Mod III	Level I				
	Extended	X			
	Disk	X			
TRS Color	Level I				
	Extended				
	Disk				
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

**CIRCLE***Format 1*

CIRCLE (x-value, y-value) ,radius [,color]  
 [,start ,end] [,aspect]

*Format 2*

CIRCLE (x-value, y-value) ,radius [,color]  
 [,ratio ] [,start ,end]

The **CIRCLE** statement draws a circle of the specified radius and color with the center at the point specified by the *x* and *y* values. The circle can be changed into an ellipse by means of the aspect parameter. If the start and end parameters are specified, an arc rather than a circle is drawn.

Parameters must appear in the order shown; if a parameter is not specified and others follow it, its absence must be indicated by two consecutive commas. Points that are off the screen are not plotted. For purposes of subsequent commands, the “last point referenced” is the center of the circle.

#### *Format 1*

This command can be issued only in graphics mode. The *x* and *y* values define the center of the circle in absolute or relative form. The radius parameter defines the *x* or *y* radius, depending on the value of aspect. If no *color* is specified, the foreground color is used; this is color 3 in medium resolution and color 1 in high resolution.

The *start* and *end* parameters specify the angle at which the circle begins and ends. They are expressed in radians. Their values must be from  $-6.283186$  to  $+6.283186$  ( $-2\pi$  to  $+2\pi$ ). If either is negative ( $-0$  is not allowed), it is treated as positive but the arc is connected to the center of the circle by a line. The starting angle can be less than the ending angle.

The *aspect* parameter defines the ratio of the *x*-axis to the *y*-axis and is used to generate ellipses. If it is less than 1, “radius” is taken as the *x*-radius; if greater than 1, “radius” is taken as the *y*-radius. The default is  $\frac{5}{6}$  for medium-resolution mode and  $\frac{5}{12}$  for high-resolution mode.

#### *Format 2*

This form of **CIRCLE** operates in much the same way as the format 1 statement. The values of the parameters and their defaults are as follows:

Parameter	Range	Default
X	0 to 255	None
Y	0 to 191	None
color	0 to 8	Foreground color
start, end	0 to 1	Start = 0, end = 1
radius	Valid x-values	None
ratio	0 to 255	1

The *ratio* is the ratio of height to width. When it is 0, the figure is a horizontal line; as the value approaches 1, the figure is an ellipse with the major axis in the *x*-direction. At 1, the figure is a circle. For values above 1

the major axis is in the  $y$ -direction and, as the value increases, the figure approaches a vertical line.

The *start* and *end* parameters determine whether a complete circle or only an arc are drawn. The circle starts at 3 o'clock, which is considered 0; 0.25 is 6 o'clock; 0.5 is 9 o'clock, and 0.75 is 12 o'clock. To use this parameter, the ratio must be specified.

### Example

To draw three concentric circles:

```
100 REM THE PROPER MODE, ETC. MUST FIRST BE
    SELECTED
120 CIRCLE (100,100), 60, 1
140 CIRCLE (100,100), 40, 1
160 CIRCLE (100,100), 25, 1
```

*See: DRAW, last point referenced, PAINT.*

### CIRCLE

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft				
IBM	Cassette				
	Disk				
	Advanced	X	1		
TRS Mod III	Level I				
	Extended				
	Disk				
TRS Color	Level I	X	2		
	Extended	X	2		
	Disk	X	2		
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

### CLEAR

*Format 1*

CLEAR [,arithmetic-expression-1] [,arithmetic-expression-2]

*Format 2*

CLEAR [arithmetic-expression] [,address]

*Format 3*

CLEAR

*Format 4*

CLR

The CLEAR statement resets variables to zero or null and, optionally, reserves memory. The current program is not erased.

*Format 1*

This form of CLEAR can also set the amount of stack space and the upper limit of memory. When executed, all previous DEFs are nullified. See the application notes for details of the parameters.

*Format 2*

This form of CLEAR can also reserve string storage and high memory. If an expression is specified, that number of bytes is reserved for string storage. If an address is specified, all memory above this address is reserved for programmer-defined use and is not taken by BASIC.

*Format 3*

This form of CLEAR resets pointers and stacks.

*Format 4*

This form of CLEAR resets pointers and stacks, cancels the dimensions of arrays, and resets the DATA pointer.

**See: HIMEM:, LOMEM:.**

CLEAR
-------

System		In	Format	Notes	Alternate Commands
APPLE	Integer	X	4	4	
	Applesoft	X	3		
	DOS	X	3		
	Microsoft	X	1	1	
IBM	Cassette	X	1	2	
	Disk	X	1	2	
	Advanced	X	1	2	
TRS Mod III	Level I				
	Extended	X	2	3	
	Disk	X	2	3	
TRS Color	Level I	X	2		
	Extended	X	2		
	Disk	X	2		
Commodore	VIC 20	X	4		
ATARI	400/800	X	4		
ANSI	Minimum				

### Notes

1. The first expression is the highest location available to BASIC; the second expression defines the stack space. The default is 256 bytes or one-eighth of the available memory, whichever is smaller. (In older versions, the first expression defined string space; the second defined the upper limit of memory.)
2. The first expression specifies the maximum number of bytes for BASIC to use for programs and data; the second expression defines the stack space. The default for stack space is 512 bytes or one-eighth of the available memory, whichever is smaller.
3. An address cannot be specified, only a value. The default is 50 bytes.
4. Does not reset arrays; cannot be issued from a program.

## CLOAD

### Format

CLOAD ["file-name"]

The CLOAD command loads a BASIC program from cassette. The previous contents of memory are erased. If a file-name is not specified, the first file encountered is loaded. Unless the tape is rewound, this is not necessarily the first file on tape.

*See: BLOAD, CLOADM, CSAVE, ENTER, LOAD, SHLOAD.*

## CLOAD

System		In	Format	Notes	Alternate Commands
APPLE	Integer				LOAD
	Applesoft				LOAD
	DOS				LOAD
	Microsoft				LOAD
IBM	Cassette				LOAD
	Disk				LOAD
	Advanced				LOAD
TRS Mod III	Level I				
	Extended	X		1	
	Disk	X		1	
TRS Color	Level I	X		2	
	Extended	X		2	
	Disk	X		2	
Commodore	VIC 20				LOAD
ATARI	400/800	X		3	
ANSI	Minimum				

*Notes*

1. The file-name must be a one-character string in quotes. As the computer searches, the names of the files encountered will be displayed in the upper right corner of the screen. When searching, only the first character of the file-name on tape is used in the comparison.
2. The file-name must be eight characters or fewer. Embedded spaces can be included. During the search a "S" is displayed at the top left of the screen; when the file has been found, the program name and an "F" are printed.
3. A file-name is not permitted. A bell rings to indicate that the PLAY button on the recorder should be pressed.

**CLOAD?***Format*

CLOAD? ["file-name"]

The CLOAD? command compares a program on tape with the program currently in memory. If no file-name is specified, the first file encountered is used. This command does *not* destroy the contents of memory.

The file-name must be one character in quotes. When searching for the file, only the first character of the file-name on tape is used in the comparison.

CLOAD? is supported only by TRS Mod III, Extended and Disk.

*See: VERIFY.*

**CLOADM**     *Format*

CLOADM file-name [,offset]

The CLOADM command loads a machine language program that has been stored on cassette using CSAVEM. The file-name must be eight characters or fewer; spaces can be embedded in the name. If a file-name is not specified, the first file encountered is loaded. Unless the tape is rewound, this is not necessarily the first file on tape.

If an offset is specified, it is added to the addresses of the program when it is loaded, thus relocating it.

CLOADM is supported only by TRS Color, Extended and Disk.

*See: BLOAD, CALL, CLOAD, CSAVEM, DLOAD, SHLOAD, SYS, USR.*

**CLOG**     *Format*

CLOG (arithmetic-expression)

The common logarithm function, CLOG, has as its value the logarithm base 10 of the expression. The expression must be greater than zero.

CLOG is supported only by ATARI BASIC.

*See: LOG.*

**CLOSE**     *Format 1*

CLOSE {file-name} ...

*Format 2*

CLOSE [[ #] file-number] ...

The CLOSE command closes the specified files or buffers. All characters in the file buffer are written to the file and the buffer is deallocated. If no file-name or number is specified, all open files are closed. If an open file that has been written to is not closed before exiting the program, data may be lost.

*See: GET, OPEN, PUT.*

CLOSE
-------

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS	X	1	1	
	Microsoft	X	2	2	
IBM	Cassette	X	2	2	
	Disk	X	2	2	
	Advanced	X	2	2	
TRS Mod III	Level I				
	Extended				
	Disk	X	2	3	
TRS Color	Level I	X	2	4, 5, 6	
	Extended	X	2	4, 5, 6	
	Disk	X	2	4, 5, 6	
Commodore	VIC 20	X	2	6	
ATARI	400/800	X	2		
ANSI	Minimum				

*Notes*

1. If an EXEC file is open, it is not closed unless explicitly written in the statement. This applies to both sequential and random files.

This command must be executed, preceded by a CONTROL-D, as part of a PRINT statement:

```
PRINT CHR$(4); "CLOSE MYFILE"
```

2. If the file is sequential, any data in the output buffer are written to the file.
3. The file-number must be between 1 and 15; it can be an arithmetic-expression.
4. In nondisk BASIC, file-number -2 is the printer, -1 is the cassette, and 0 is the screen or keyboard.
5. The number sign (#) cannot be used in nondisk BASIC; in Disk BASIC it must be used.
6. The number sign (#) cannot be used in the statement.

**CLR**

*See: CLEAR.*

**CLS**

*Format*

CLS (color)

The CLS statement clears the screen and, optionally, sets it to the indicated color.

*See: color codes, HOME.*

CLS
-----

System		In	Format	Notes	Alternate Commands
APPLE	Integer				CALL -936
	Applesoft				HOME
	DOS				
	Microsoft				
IBM	Cassette	X		1	
	Disk	X		1	
	Advanced	X		1	
TRS Mod III	Level I	X			
	Extended	X		2	
	Disk	X		2	
TRS Color	Level I	X		3	
	Extended	X		3	
	Disk	X		3	
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

### Notes

1. A color cannot be specified. The cursor is moved to the "home" position. In text mode this is the upper left corner; in graphics mode this is the center of the screen, which then becomes the "last point referenced." (For medium resolution, this is point 160,100; for high resolution, it is point 320,100.) In graphics mode the screen is set to the background color.
2. Turns off all graphics blocks and moves the cursor to the upper leftmost position; a color cannot be specified.
3. The color can be designated by an expression between 0 and 8, inclusive. When necessary, the expression is truncated to an integer. If omitted, the screen is set to green.

## CMD

### Format 1

CMD file-number

*Format 2*

CMD "letter" [,options]

*Format 1*

This form of the CMD statement sends output that would normally go to the screen to another device, or to a file on tape or disk.

The file-number must be between 1 and 15; it specifies the file (not the device), to which the output goes. To return output to the screen, close the file.

*Format 2*

This form of CMD is unique to TRS Mod III DOS. The valid letters and their effects are as follows:

**CMD "A"**

Returns to DOS with an error message.

**CMD "B", "ON" | "OFF"**

Enables or disables the break key. When disabled, the break key is ignored except during cassette, printer, or serial input/output. The key remains disabled even after the program has ended. Returning to DOS with CMD "S" or CMD "I" reenables the break key.

**CMD "C" [,R] [,S]**

Compresses a program. If R is specified, all remarks are deleted. If S is specified, all unnecessary spaces are deleted. Both may be specified at the same time.

**CMD "D:drive"**

Displays the directory of the specified drive. Only unprotected files are listed. A drive must be specified; there is no default.

**CMD "E"**

Displays the last DOS error message or "NO ERROR FOUND."

**CMD "I", command**

Executes the specified DOS command or a Z-80 program. Control then returns to the BASIC program, if it has not been destroyed. This command also enables the break key. The command can be in a string variable; if it is a literal, it must be in quotes.

**CMD “J”, source-string, destination-string**

Converts a date from one format to another. The date in the source-string is converted and left in the destination-string. The two forms that the date can have are mm/dd/yy and –yy/ddd, where the first format is the standard month, day, and year, and the second format is the so-called Julian representation, consisting of the year and the day of the year, from 1 to 365 (or 366).

**CMD “L”, program-name**

Loads a Z-80 routine into memory. If the BASIC program is not destroyed, control returns to it after the operation. The program-name can be in a string variable; if it is a literal, it must be in quotes.

**CMD “O”, count, array-name(starting-element)**

Sorts a one-dimensional string array. The count specifies the number of items to be sorted; it must be an integer variable. The array is sorted from the designated starting element to the end.

**CMD “P”, string-variable**

Checks the status of the printer. The value returned to the string-variable is a status byte. If the left 4 bits are 0011, the printer is ready. (The values of the other 4 bits depend on the particular printer used; refer to the printer manual.)

**CMD “R”**

Turns on the real-time clock display. The time of day is displayed in the upper right corner of the screen. It is updated each second. The clock is turned off during cassette and disk input/output operations.

**CMD “T”**

Turns off the real-time clock display.

**CMD “S”**

Returns control to DOS, enabling the break key. (See CMD “B”.)

**CMD “X”, word**

Finds all occurrences of the specified word. To search for a BASIC reserved word, it must not be in quotes in the command. All instances of the word, except those in literals and remarks, will be listed. To search for anything else, the word must be in quotes. All instances of it, except where it is used as a BASIC reserved word, will be listed.

**CMD “Z”, “ON” | “OFF”**

Enables or disables screen output to printer. When enabled, any output to the screen will also go to the printer, and any printer output will also go to the screen. The printer must be on-line when the command is given.

CMD
-----

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft				
IBM	Cassette				
	Disk				
	Advanced				
TRS Mod III	Level I				
	Extended				
	Disk	X	2		
TRS Color	Level I				
	Extended				
	Disk				
Commodore	VIC 20	X	1		
ATARI	400/800				
ANSI	Minimum				

**COLOR***Format 1*

COLOR = arithmetic-expression

*Format 2*

COLOR [background-color] [,palette]

*Format 3*

COLOR [foreground-color] [,background-color] [,border-color]

*Format 4*

COLOR foreground-color, background-color

*Format 5*

COLOR arithmetic-expression

The COLOR statement sets the foreground and background screen colors and, optionally, the border color.

*Format 1*

This form of COLOR sets the screen color for low-resolution graphics. The value of the expression can be between 0 and 255; it is taken as an integer, modulo 16. In high-resolution graphics mode, this command is ignored.

*Format 2*

This form of COLOR is used for medium-resolution graphics. Any parameter can be an arithmetic-expression. If any parameter is omitted, it assumes its old value.

The palette can be between 0 and 255; if it is even, palette 0 is selected; if odd, palette 1. (Palette 0 is green/red/brown; palette 1 is cyan/magenta/white.) The background color can be between 0 and 15.

*Format 3*

This form of COLOR is used in text mode. For the color/graphics display, the foreground color can be between 0 and 31. (Values of 16 to 31 cause colors 0 to 15 to blink.) The background color can be between 0 and 7; the border color, between 0 and 15.

For the monochrome display, the only background colors are 0 (black) and 7 (white). For the foreground colors, in addition to 0 and 7, 15 (high-intensity white) and 1 (underlined character with white foreground) can also be used. If 16 is added to any of these values, the result is a blinking color. A border color cannot be specified. If any parameter is omitted, its old value is used.

*Format 4*

Either color parameter can be an expression between 0 and 8, inclusive.

*Format 5*

This form of COLOR determines the color of subsequent PLOT and DRAWTO statements.

The actual color depends on the value in the color register that corresponds to the value of the expression for the mode being used. The expression must have a value between 0 and 255; nonintegers are rounded to the nearest integer. The following chart shows how this value is interpreted.

Mode	Value of Expression	Selects Color Register	Default Color
0, 1, and 2, and all TEXT windows	0 to 255	Value determines character and color	
3, 5, and 7	0	4	Black
	1	0	Orange
	2	1	Light green
	3	2	Dark blue
4 and 6	0	4	Black
	1	0	Orange
8	0	2	Dark blue
	1	1	Light green

For modes 3 to 8, if the value is over 1 or 3, the lower 1 or 2 bits is taken.

*See: color codes, GR, GRAPHICS, HCOLOR, HGR, PLOT, SCRN, SETCOLOR.*

COLOR
-------

System		In	Format	Notes	Alternate Commands
APPLE	Integer	X	1		
	Applesoft	X	1	1	
	DOS	X	1	1	
	Microsoft	X	1	2	
IBM	Cassette	X	2, 3		
	Disk	X	2, 3		
	Advanced	X	2, 3		
TRS Mod III	Level I				
	Extended				
	Disk				
TRS Color	Level I				
	Extended	X	4		
	Disk	X	4		
Commodore	VIC 20				
ATARI	400/800	X	5		SETCOLOR
ANSI	Minimum				

*Notes*

1. COLOR is parsed as a reserved word only if the next nonspace character is an equal sign (=).
2. If a color is not specified in GR, it is set to 0.

**color codes**

The following chart defines the color associated with the numeric codes for the various systems.

Code	ATARI	IBM (1)	Apple	Micro- soft (2)	TRS (3)	Commo- dore
0	Gray	Black	Black	Black	Black	
1	Light orange	Blue	Magenta	Green	Green	Black
2	Orange	Green	Dark blue	Violet	Yellow	White
3	Red-orange	Cyan	Purple	White	Blue	Red
4	Pink	Red	Dark green	Black	Red	Cyan
5	Purple	Magenta	Gray	Orange	Buff	Purple
6	Purple-blue	Brown	Med. Blue	Blue	Cyan	Green
7	Blue	White	Light blue	White	Magenta	Blue
8	Blue	Gray	Brown	Black-1	Orange	Yellow
9	Light blue	Light blue	Orange	White-1		
10	Aqua	Light green	Gray	Black-2		
11	Green-blue	Light cyan	Pink	White-2		
12	Green	Light red	Green	Reverse		
13	Yellow-green	Light magenta	Yellow			
14	Orange-green	Yellow	Aqua			
15	Light orange	High-white	White			

*Notes*

1. Colors 8 to 14 are lighter versions of 0 to 6.
2. Black-1 and -2 and white-1 and -2 are thicker lines.
3. If in four-color mode, 4 is subtracted from values of 5 to 8.

**COM***Format*

COM (channel) ON|OFF|STOP

The COM command enables or disables the trapping of communications activity to the indicated communications adapter.

COM ON allows trapping by an ON COM ... GOSUB statement; COM OFF disables this trapping. COM STOP disables trapping, but if any activity occurs it is remembered and a trap occurs as soon as an ON COM statement is executed. The channel must be an integer with a value of 1 or 2.

COM is supported only by IBM Advanced BASIC.

*See: ON COM, trap.*

**command** A command is a directive to the system to do something with a program or part of a program.

*See: function, statement.*

**command level** Command level refers to that mode of BASIC in which direct statements (that is, those without a line number) can be executed. It is also called “immediate mode” and “monitor level.”

For example, if one types 100 X = 123, the statement is not immediately executed; rather, it is stored and executed only when the program is run. On the other hand, typing X = 123 causes the variable X to be immediately assigned the value 123.

**COMMON** *Format*

COMMON {variable,...}

The COMMON statement defines which variables are available to a chained program. A given variable cannot appear in more than one COMMON statement in any one program. To indicate an array, use the array's name followed by an empty pair of parentheses.

For an example of the use of the COMMON statement in a program, see “CHAIN.”

*Example*

100 COMMON A,B\$,C(),D,ES

*See: CHAIN.*

COMMON

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft	X			
IBM	Cassette				
	Disk	X			
	Advanced	X			
TRS Mod III	Level I				
	Extended				
	Disk				
TRS Color	Level I				
	Extended				
	Disk				
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

**com-  
plement**

*See: one's complement, two's complement.*

**concate-  
nation**

Concatenation is the operation of joining two strings together. The concatenation operator is (+). If  $X\$ = \text{"ABC"}$  and  $Y\$ = \text{"DEF"}$ , the result of concatenating the two, ( $X\$ + Y\$$ ), is the string "ABCDEF".

In Apple and ATARI BASIC, concatenation of strings is performed by setting the  $(N + 1)$ st element of an  $N$ -element string equal to the value to be concatenated. For example, assume that string  $A\$$  is defined as being 10 characters long, and that its present contents are "ABC". The statement  $A\$(4) = \text{"DEF"}$  results in the new value of  $A\$$  being "ABCDEF". In general,  $X\$(\text{LEN}(X\$) + 1) = Y\$$  concatenates  $Y\$$  to  $X\$$ .

**CONT**

*Format*

**CONT**

CONT is used to continue program operation after a stop, end, or program break. This is an immediate mode instruction only, and cannot appear in a program.

When a program has been interrupted, the values of variables can be changed, but if any part of the *program* has been changed, CONT cannot be used to continue; the program must be restarted, or entered by a GO TO.

*See: END, STOP.*

CONT

System		In	Format	Notes	Alternate Commands
APPLE	Integer	X		1	
	Applesoft	X		2	
	DOS	X		2	
	Microsoft	X		3	
IBM	Cassette	X		3	
	Disk	X		3	
	Advanced	X		3	
TRS Mod III	Level I	X			
	Extended	X			
	Disk	X			
TRS Color	Level I	X			
	Extended	X			
	Disk	X			
Commodore	VIC 20	X		4	
ATARI	400/800	X			
ANSI	Minimum				

### Notes

1. Must be spelled "CON".
2. If a CONTROL-C is used to interrupt an INPUT statement, use of CONT will cause an error message.
3. If a break occurs during an INPUT statement, the prompt is reissued and the statement begins all over again.
4. If the program has halted due to an error, or if an error was caused while in immediate mode, use of CONT will give an error.

### conversion tables

The following table shows the equivalent values for decimal, binary, hexadecimal, and octal numbers that can be contained in one byte.

## CONVERSION TABLES

DEC.	BINARY	HEX.	OCT.
0	00000000	00	000
1	00000001	01	001
2	00000010	02	002
3	00000011	03	003
4	00000100	04	004
5	00000101	05	005
6	00000110	06	006
7	00000111	07	007
8	00001000	08	010
9	00001001	09	011
10	00001010	0A	012
11	00001011	0B	013
12	00001100	0C	014
13	00001101	0D	015
14	00001110	0E	016
15	00001111	0F	017
16	00010000	10	020
17	00010001	11	021
18	00010010	12	022
19	00010011	13	023
20	00010100	14	024
21	00010101	15	025
22	00010110	16	026
23	00010111	17	027
24	00011000	18	030
25	00011001	19	031
26	00011010	1A	032
27	00011011	1B	033
28	00011100	1C	034
29	00011101	1D	035
30	00011110	1E	036
31	00011111	1F	037
32	00100000	20	040
33	00100001	21	041
34	00100010	22	042
35	00100011	23	043
36	00100100	24	044
37	00100101	25	045
38	00100110	26	046
39	00100111	27	047
40	00101000	28	050
41	00101001	29	051
42	00101010	2A	052
43	00101011	2B	053
44	00101100	2C	054
45	00101101	2D	055
46	00101110	2E	056
47	00101111	2F	057
48	00110000	30	060
49	00110001	31	061
50	00110010	32	062
51	00110011	33	063
52	00110100	34	064
53	00110101	35	065
54	00110110	36	066
55	00110111	37	067
56	00111000	38	070
57	00111001	39	071
58	00111010	3A	072
59	00111011	3B	073
60	00111100	3C	074
61	00111101	3D	075
62	00111110	3E	076
63	00111111	3F	077
64	01000000	40	100

DEC.	BINARY	HEX.	OCT.
65	01000001	41	101
66	01000010	42	102
67	01000011	43	103
68	01000100	44	104
69	01000101	45	105
70	01000110	46	106
71	01000111	47	107
72	01001000	48	110
73	01001001	49	111
74	01001010	4A	112
75	01001011	4B	113
76	01001100	4C	114
77	01001101	4D	115
78	01001110	4E	116
79	01001111	4F	117
80	01010000	50	120
81	01010001	51	121
82	01010010	52	122
83	01010011	53	123
84	01010100	54	124
85	01010101	55	125
86	01010110	56	126
87	01010111	57	127
88	01011000	58	130
89	01011001	59	131
90	01011010	5A	132
91	01011011	5B	133
92	01011100	5C	134
93	01011101	5D	135
94	01011110	5E	136
95	01011111	5F	137
96	01100000	60	140
97	01100001	61	141
98	01100010	62	142
99	01100011	63	143
100	01100100	64	144
101	01100101	65	145
102	01100110	66	146
103	01100111	67	147
104	01101000	68	150
105	01101001	69	151
106	01101010	6A	152
107	01101011	6B	153
108	01101100	6C	154
109	01101101	6D	155
110	01101110	6E	156
111	01101111	6F	157
112	01110000	70	160
113	01110001	71	161
114	01110010	72	162
115	01110011	73	163
116	01110100	74	164
117	01110101	75	165
118	01110110	76	166
119	01110111	77	167
120	01111000	78	170
121	01111001	79	171
122	01111010	7A	172
123	01111011	7B	173
124	01111100	7C	174
125	01111101	7D	175
126	01111110	7E	176
127	01111111	7F	177
128	10000000	80	200
129	10000001	81	201

DEC.	BINARY	HEX.	OCT
130	10000010	82	202
131	10000011	83	203
132	10000100	84	204
133	10000101	85	205
134	10000110	86	206
135	10000111	87	207
136	10001000	88	210
137	10001001	89	211
138	10001010	8A	212
139	10001011	8B	213
140	10001100	8C	214
141	10001101	8D	215
142	10001110	8E	216
143	10001111	8F	217
144	10010000	90	220
145	10010001	91	221
146	10010010	92	222
147	10010011	93	223
148	10010100	94	224
149	10010101	95	225
150	10010110	96	226
151	10010111	97	227
152	10011000	98	230
153	10011001	99	231
154	10011010	9A	232
155	10011011	9B	233
156	10011100	9C	234
157	10011101	9D	235
158	10011110	9E	236
159	10011111	9F	237
160	10100000	A0	240
161	10100001	A1	241
162	10100010	A2	242
163	10100011	A3	243
164	10100100	A4	244
165	10100101	A5	245
166	10100110	A6	246
167	10100111	A7	247
168	10101000	A8	250
169	10101001	A9	251
170	10101010	AA	252
171	10101011	AB	253
172	10101100	AC	254
173	10101101	AD	255
174	10101110	AE	256
175	10101111	AF	257
176	10110000	B0	260
177	10110001	B1	261
178	10110010	B2	262
179	10110011	B3	263
180	10110100	B4	264
181	10110101	B5	265
182	10110110	B6	266
183	10110111	B7	267
184	10111000	B8	270
185	10111001	B9	271
186	10111010	BA	272
187	10111011	BB	273
188	10111100	BC	274
189	10111101	BD	275
190	10111110	BE	276
191	10111111	BF	277
192	11000000	C0	300
193	11000001	C1	301
194	11000010	C2	302

DEC.	BINARY	HEX.	OCT.
195	11000011	C3	303
196	11000100	C4	304
197	11000101	C5	305
198	11000110	C6	306
199	11000111	C7	307
200	11001000	C8	310
201	11001001	C9	311
202	11001010	CA	312
203	11001011	CB	313
204	11001100	CC	314
205	11001101	CD	315
206	11001110	CE	316
207	11001111	CF	317
208	11010000	D0	320
209	11010001	D1	321
210	11010010	D2	322
211	11010011	D3	323
212	11010100	D4	324
213	11010101	D5	325
214	11010110	D6	326
215	11010111	D7	327
216	11011000	D8	330
217	11011001	D9	331
218	11011010	DA	332
219	11011011	DB	333
220	11011100	DC	334
221	11011101	DD	335
222	11011110	DE	336
223	11011111	DF	337
224	11100000	E0	340
225	11100001	E1	341
226	11100010	E2	342
227	11100011	E3	343
228	11100100	E4	344
229	11100101	E5	345
230	11100110	E6	346
231	11100111	E7	347
232	11101000	E8	350
233	11101001	E9	351
234	11101010	EA	352
235	11101011	EB	353
236	11101100	EC	354
237	11101101	ED	355
238	11101110	EE	356
239	11101111	EF	357
240	11110000	F0	360
241	11110001	F1	361
242	11110010	F2	362
243	11110011	F3	363
244	11110100	F4	364
245	11110101	F5	365
246	11110110	F6	366
247	11110111	F7	367
248	11111000	F8	370
249	11111001	F9	371
250	11111010	FA	372
251	11111011	FB	373
252	11111100	FC	374
253	11111101	FD	375
254	11111110	FE	376
255	11111111	FF	377

**COPY***Format***COPY file-specification-1 TO file-specification-2**

The file designated by file-specification-1 is copied to the disk specified in file-specification-2 and is given the name in that specification. If file-specification-2 specifies only a drive, the file is copied to that drive with the same name as in file-specification-1.

COPY
------

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft				
IBM	Cassette				
	Disk				
	Advanced				
TRS Mod III	Level I				
	Extended				
	Disk	X		1	
TRS Color	Level I				
	Extended				
	Disk	X		2	
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

*Notes*

1. "TO" cannot be used; the names are separated by a space. The statement COPY /extension: drive-number copies all files with the specified extension to the indicated drive. The file-specification cannot be in quotes.
2. The file-specification must have an extension and must be in quotes. When executed, memory is erased.

**COS***Format***COS (arithmetic-expression)**

The cosine function, COS, has as its value the cosine of the angle specified by the expression, which is interpreted to be in radians. To find the cosine of an angle expressed in degrees, use COS (X \* .01745329).

If the COS function is not implemented, the cosine of an angle, expressed in radians, can be calculated by the series

$$\cos X = 1 - \frac{X^2}{2!} + \frac{X^4}{4!} - \frac{X^6}{6!} + \frac{(-1)^n X^{2n}}{(2n)!}$$

*Example*

```
100 XR = 45 * .0174533
120 Y = COS(XR)
140 PRINT COS(.5235988), Y, COS(-XR)
```

**Output**

```
.8660254    .7071068    .7071068
```

(0.5235988 radian = 30 degrees.)

*See: ATN, DEG, RAD, SIN, TAN, trigonometric functions.*

COS
-----

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft	X			
	DOS	X			
	Microsoft	X			
IBM	Cassette	X			
	Disk	X			
	Advanced	X			
TRS Mod III	Level I				
	Extended	X			
	Disk	X			
TRS Color	Level I				
	Extended	X			
	Disk	X			
Commodore	VIC 20	X			
ATARI	400/800	X		1	
ANSI	Minimum	X			

*Note*

1. The argument can be in degrees or radians, depending on whether DEG or RAD was executed.

**CSAVE***Format*

CSAVE "program-name" [,A]

The CSAVE command stores the program currently in memory on cassette. If an "A" is specified, the program is stored in ASCII format. The program-name must be eight or fewer characters and must be in quotes.

*See: BSAVE, CSAVEM, ENTER, LIST, LOAD, SAVE.*

CSAVE
-------

System		In	Format	Notes	Alternate Commands
APPLE	Integer				SAVE
	Applesoft				SAVE
	DOS				SAVE
	Microsoft				SAVE
IBM	Cassette				SAVE
	Disk				SAVE
	Advanced				SAVE
TRS Mod III	Level I				
	Extended	X		1	
	Disk	X		1	
TRS Color	Level I	X			
	Extended	X			
	Disk	X			
Commodore	VIC 20				SAVE
ATARI	400/800	X		2	
ANSI	Minimum				

*Notes*

1. The program-name can be only one character long.
2. A program name cannot be specified. The resident program is stored on tape in tokenized form. Two bells ring to tell when to press Play/Record. This executes faster than SAVE as it uses short inter-record gaps.

**CSAVEM***Format*

CSAVEM "program-name", starting-address, ending-address,  
execution-address

The CSAVEM command saves a machine language program on cassette. The program saved is between the starting and ending address. When loaded, control is passed to the transfer address. All addresses are interpreted to be hexadecimal.

CSAVEM is supported only by TRS Color, Extended and Disk.

*See: BSAVE, CSAVE, SAVE.*

## CSNG

### Format

CSNG (arithmetic-expression)

The convert-to-single-precision function, CSNG, has as its value the single-precision representation of the expression. If the expression is double-precision, 4/5 rounding is used. That is, if a single-precision number has N digits, a value of 5 is added to the (N + 1)st digit of the argument. The first N digits of the result are then taken as the single-precision value.

### Example

```
100 DEFDBL D
120 D = 12345.67890123
140 X = CSNG(D)
160 PRINT X, CSNG(-1.234567890123)
```

### Output

```
12345.68      -1.234568
```

*See: CDBL, CINT.*

CSNG
------

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft	X			
IBM	Cassette	X			
	Disk	X			
	Advanced	X			
TRS Mod III	Level I				
	Extended	X		1	
	Disk	X		1	
TRS Color	Level I				
	Extended				
	Disk				
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

*Note*

1. Returned as a six-digit number.

**CSRLIN***Format*

CSRLIN

The CSRLIN function returns as its value the vertical coordinate of the cursor on the active page. The value returned can range from 1 to 25, and corresponds to the current line or row. The active page is not necessarily the one that is being displayed on the screen.

*Example*

```
100 Y = 0
120 LOCATE (15,10)
140 Y = CSRLIN
160 PRINT Y
```

**Output**

15

CSRLIN is supported only by IBM BASIC (all levels).

*See: LOCATE, POS, POSITION, VPOS.*

**cursor**

The cursor is the symbol that shows where on the screen the next character to be typed in or printed out is displayed. The form that the cursor takes varies among implementations. Some systems provide the ability for the user to change the form of the cursor.

*See: LOCATE, POSITION.*

**CVD***Format*

CVD (string-expression)

The CVD function has as its value the numeric value of the argument expressed as a double-precision number. This is the inverse function of MKD\$.

The expression in the argument must be 8 bytes. It is generally the result of putting a double-precision number through the MKD\$ function, prior to fielding it. If the expression is greater than 8 bytes, only the first 8 bytes are used.

*Example*

```
100 DEFDBL Y
120 X$ = MKD$(12345.67890123)
140 Y = CVD(X$)
160 PRINT Y
```

**Output**

12345.67890123

*See: FIELD, GET, LSET, MKD\$, PUT, RSET.*

CVD

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft	X			
IBM	Cassette				
	Disk	X			
	Advanced	X			
TRS Mod III	Level I				
	Extended				
	Disk	X			
TRS Color	Level I				
	Extended				
	Disk				
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

**CVI**

*Format*

CVI (string-expression)

The CVI function has as its value the numeric value of the argument expressed as an integer. This is the inverse function of MKI\$.

The expression in the argument must be 2 bytes. It is generally the result of putting an integer through the MKI\$ function, prior to fielding it. If the expression is greater than 2 bytes, only the first 2 bytes are used.

*Example*

```
100 DEFINT Y
120 X$ = MKI$(12345)
140 Y = CVI(X$)
160 PRINT Y
```

**Output**

12345

*See: FIELD, GET, LSET, MKI\$, PUT, RSET.*

CVI
-----

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft	X			
IBM	Cassette				
	Disk	X			
	Advanced	X			
TRS Mod III	Level I				
	Extended				
	Disk	X			
TRS Color	Level I				
	Extended				
	Disk				
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

**CVN***Format*

CVN (string-expression)

The CVN function has as its value the numeric value of the argument. This is the inverse function of MKN\$. The expression in the argument must be a 5-byte string that was created by MKN\$.

*Example*

```
100 X$ = MKN$(12345)
120 Y = CVN(X$)
140 PRINT Y
```

**Output**

12345

CVN is supported only by TRS Color DOS.

*See: FIELD, GET, LSET, MKN\$, PUT, RSET.*

**CVS***Format*

CVS (string-expression)

The CVS function has as its value the numeric value of the argument expressed as a single precision number. This is the inverse function of MKS\$.

The expression in the argument must be 4 bytes. It is generally the result of putting a single-precision number through the MKS\$ function prior to fielding it. If the expression is greater than 4 bytes, only the first 4 bytes are used.

*Example*

```
100 X$ = MKS$(12345.678)
120 Y = CVS(X$)
160 PRINT Y
```

**Output**

12345.68

***See: FIELD, GET, LSET, MKS\$, PUT, RSET.***

CVS

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft	X			
IBM	Cassette				
	Disk	X			
	Advanced	X			
TRS Mod III	Level I				
	Extended				
	Disk	X			
TRS Color	Level I				
	Extended				
	Disk				
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

**DATA***Format*

DATA {value,}...

The DATA statement specifies one or more values to be input to the program via a READ statement.

DATA statements can occur anywhere in the program. Conceptually, they are considered to be all together, in order from lowest to highest line number. The first READ in the program accesses the first value of the first DATA statement; the next READ accesses the second value of the first DATA statement or, if there is no second value, the first value of the next DATA statement; and so on.

The values must be numeric or string constants; variables or expressions are not allowed. Commas must be used to separate the elements; if two commas follow one another with no intervening characters, a value of zero or null, whichever is appropriate to the variable being given a value, is used.

String literals do not have to have quotes around them. If a string element does not begin with a quote, then leading and trailing spaces are not taken as part of the string, and a comma or colon in the string will function as a delimiter. However, a quote can appear inside the string.

If a string element begins with a quote, then a comma, colon, or significant leading or trailing spaces can be part of the string. But, a quote cannot be in the string. For an example of the use of the DATA statement, see "READ."

*See: READ, RESTORE.*

DATA
------

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft	X		1	
	DOS	X		1	
	Microsoft	X			
IBM	Cassette	X			
	Disk	X			
	Advanced	X			
TRS Mod III	Level I	X			
	Extended	X			
	Disk	X			
TRS Color	Level I	X			
	Extended	X			
	Disk	X			
Commodore	VIC 20	X			
ATARI	400/800	X			
ANSI	Minimum	X			

*Note*

1. CONTROL-X or CONTROL-M cannot be part of any element in the statement; a CONTROL-C can be part of an element, however.

**DATE\$***Format*

DATE\$

The DATE\$ function has as its value the current date in the form

mm-dd-yyyy

where “mm” is the month, “dd” the date, and “yyyy” the year.

DATE\$ also functions as a variable, in that it can be assigned a value. (In fact, this has the same value as the system date which is assigned at startup.) It can be assigned a value in any of the forms

```
mm-dd-yy
mm-dd-yyyy
mm/dd/yy
mm/dd/yyyy
```

When two digits are used for the year, 19xx is assumed.

*Example*

```
100 PRINT DATE$
```

**Output**

```
08-26-1983
```

*Example*

```
100 Y$ = "8-27-83"
120 DATE$ = Y$      (or DATE$ = "8-27-83")
140 PRINT DATE$
```

**Output**

```
08-27-1983
```

DATE\$ is supported only by Disk and Advanced IBM BASIC.

**decrement** A decrement is either a negative value that is added to, or a positive value that is subtracted from, a counter or variable. The net result is to reduce the value of the counter or variable.

*See: increment.*

## DEF FN *Format*

DEF FN function-name [(dummy-variable) ...] = numeric-expression

DEF FN is used to define a numeric function. When this function is used in a program, the function is equivalent to the numeric expression but with real values substituted for the dummy variables. A function can be redefined anywhere in the program. The function name can be one or two letters.

### *Example*

The following function has as its value the fourth root of the argument.

```
100 DEF FNR4(X) = SQR(SQR(X))
120 A = 4096
140 Y = FNR4(A)
160 PRINT Y, FNR4(1.23456E-08)
```

### **Output**

```
8      1.054091E-02
```

### *Example*

The following defines the modulus function, which is a function of two variables.

```
100 DEF FNMD (A,B) = (INT(A) - INT(INT(A)/INT(B)) * INT(B))
120 X = 12345 : Y = 123
140 Z = FNMD(57,7)
160 PRINT Z, FNMD(X,Y)
```

### **Output**

```
1      45
```

DEF FN

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft	X		1,2,3,6	
	DOS	X		1,2,3,6	
	Microsoft	X		2, 5, 7	
IBM	Cassette	X		2,4,5,7	
	Disk	X		2,4,5,7	
	Advanced	X		2,4,5,7	
TRS Mod III	Level I				
	Extended				
	Disk	X			
TRS Color	Level I				
	Extended	X		6	
	Disk	X		6	
Commodore	VIC 20	X		6	
ATARI	400/800				
ANSI	Minimum	X			

*Notes*

1. Only the first two characters of the function-name are significant.
2. The definition can be only one line long.
3. Variables to the left of the equal sign (=) cannot be integers.
4. Recursive definitions are permitted.
5. The function-name can be any valid name up to 40 characters.
6. Only one variable can be specified in the definition.
7. The function can also be a string function. In this case the function-name must be a string-variable name.

**DEF SEG***Format*

DEF SEG [=address]

The DEF SEG command is used to define the current segment of storage. Subsequent BLOAD, BSAVE, CALL, PEEK, POKE, VARPTR, or USR statements define actual physical addresses as offsets into this segment.

Address is a numeric-expression between 0 and 65535. It should be a multiple of 16. If no address is specified, the current segment is defined as BASIC's data segment (which is also the initial default.)

DEF and SEG must be separated by at least one space.

DEF SEG is supported only by IBM BASIC (all levels).

## DEF USR *Format*

DEF USR[digit] = address

The DEF USR statement defines the starting address of a machine language subroutine. This statement must appear before the routine is invoked by the USR statement.

The address can be in decimal or hexadecimal. The digit must be between 0 and 9. If no digit is specified, 0 is assumed. The digit and USR cannot have a space between them. The starting address can be redefined.

*See: CALL, USR.*

### DEF USR

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft	X			
IBM	Cassette	X		1, 3	
	Disk	X		1, 3	
	Advanced	X		1, 3	
TRS Mod III	Level I				
	Extended				
	Disk	X		2	
TRS Color	Level I				
	Extended	X		3	
	Disk	X		3	
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

### Notes

1. The address is an offset that is added to the value of the current segment.
2. For addresses over 32767, use the address minus 65536.
3. The address can be between 0 and 65535, inclusive.

## default

A default is that value, device, or specification that the computer assumes if nothing is explicitly stated. For example, in a disk operation, if the default drive is 0, all disk operations are done to that disk unless some other disk is explicitly designated in the statement.

Some implementations allow the user to set up default specifications, others have only predefined ones. Often, once a parameter has been assigned, it becomes the default.

**DEFDBL***Format*

DEFDBL {letter [ – letter]} ...

The DEFDBL statement causes any program variable that begins with one of the specified letters to be treated as a double-precision number. This obviates the need to define a double-precision variable explicitly by the type declaration “#”. However, when used, a type declaration always takes precedence over the DEFDBL statement.

*Example*

The Statement:	Defines as Double Precision:
DEFDBL A	All variables beginning with an A
DEFDBL A – D	All variables beginning with A, B, C, or D
DEFDBL A,B,I – K	All variables beginning with A, B, I, J, or K

*See: DEFINT, DEFSNG, DEFSTR, type declaration.*

## DEFDBL

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft	X			
IBM	Cassette	X			
	Disk	X			
	Advanced	X			
TRS Mod III	Level I				
	Extended	X			
	Disk	X			
TRS Color	Level I				
	Extended				
	Disk				
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

**DEFINT***Format*

DEFINT {letter [– letter]} ...

The DEFINT statement causes any program variable that begins with one of the specified letters to be treated as an integer. This obviates the need to define an integer explicitly by the type declaration “%”. However, when used, a type declaration always takes precedence over the DEFINT statement.

*Example*

The Statement:	Defines as an Integer:
DEFINT A	All variables beginning with an A
DEFINT A – D	All variables beginning with A, B, C, or D
DEFINT A,B,I – K	All variables beginning with A, B, I, J, or K

*See: DEFDBL, DEFSNG, DEFSTR, type declaration.*

**DEFINT**

System	In	Format	Notes	Alternate Commands
APPLE	Integer			
	Applesoft			
	DOS			
	Microsoft	X		
IBM	Cassette	X		
	Disk	X		
	Advanced	X		
TRS Mod III	Level I			
	Extended	X		
	Disk	X		
TRS Color	Level I			
	Extended			
	Disk			
Commodore	VIC 20			
ATARI	400/800			
ANSI	Minimum			

**DEFSNG***Format*

DEFSNG {letter [– letter]} ...

The DEFSNG statement specifies that any program variable that begins with one of the specified letters will be treated as a single-precision number.

A single-precision number can also be designated by the type declaration “!”. (In general, the default for a variable is single precision.) When used, a type declaration always takes precedence over the DEFSNG statement.

*Example*

The Statement:	Defines as Single Precision:
DEFSNG A	All variables beginning with an A
DEFSNG A – D	All variables beginning with an A, B, C, or D
DEFSNG A,B,I – K	All variables beginning with A, B, I, J, or K

*See: DEFDBL, DEFINT, DEFSTR, type declaration.*

DEFSNG

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft	X			
IBM	Cassette	X			
	Disk	X			
	Advanced	X			
TRS Mod III	Level I				
	Extended	X			
	Disk	X			
TRS Color	Level I				
	Extended				
	Disk				
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

## DEFSTR

*Format*

DEFSTR {letter [– letter]}...

The DEFSTR statement causes any program variable that begins with one of the specified letters to be treated as a string variable. This obviates the need to define a string variable explicitly by the type declaration “\$”. However, when used, a type declaration always takes precedence over the DEFSTR statement.

*Example*

The Statement:	Defines as String Variables:
DEFSTR A	All variables beginning with an A
DEFSTR A – D	All variables beginning with A, B, C, or D
DEFSTR A,B,I – K	All variables beginning with A, B, I, J, or K

*See: DEFDBL, DEFINT, DEFSNG, type declaration.*

DEFSTR
--------

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft	X			
IBM	Cassette	X			
	Disk	X			
	Advanced	X			
TRS Mod III	Level I				
	Extended	X			
	Disk	X			
TRS Color	Level I				
	Extended				
	Disk				
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

**DEG***Format***DEG**

The DEG command causes the trigonometric functions to interpret their arguments in degrees rather than radians. This stays in effect until a RAD is executed. The default at startup is to interpret arguments in radians.

DEG is supported only by ATARI BASIC.

*See: ATN, COS, RAD, SIN, TAN, trigonometric functions.*

**DELETE***Format 1*

DEL [line-number-1] [– line-number-2]

*Format 2*

DELETE [line-number-1] [— line-number-2]

*Format 3*

DELETE file-name [,Ddrive] [,Sslot] [,Vvolume]

*Formats 1 and 2*

These forms of the DELETE command erase one or more program lines from memory and then return to the command level. The two forms operate identically, the only difference being in the spelling of the keyword.

If only line-number-1 is specified, it is deleted. If two line numbers are specified, all lines from line-number-1 to line-number-2, inclusive, are deleted. If only the hyphen and line-number-2 are specified, all lines from the first line in the program to line-number-2, inclusive, are deleted. Line-number-2, when specified, must be the number of a line that actually exists in the program.

After the command has executed, control returns to the command level.

*Format 3*

This form of DELETE erases the specified file. If the file was open, it is closed before being deleted.

This command must be executed, preceded by a CONTROL-D, as part of a PRINT statement:

PRINT CHR\$(4); "DELETE MYFILE, D1, S6"

The allowable values for parameters and their defaults are shown below. Once a value has been changed, the last value specified is the default.

Parameter	Range	Default
drive	1 or 2	Drive 1
slot	1 to 7	Slot from which DOS was booted
volume	1 to 254	Volume from which DOS was booted

*See: CHAIN, MERGE.*

## DELETE

System		In	Format	Notes	Alternate Commands
APPLE	Integer	X	1	1, 2	
	Applesoft	X	1	1,2,3,4	
	DOS	X	3		
	Microsoft	X	1, 2	5, 7	
IBM	Cassette	X	2	7	
	Disk	X	2	7	
	Advanced	X	2	7	
TRS Mod III	Level I				
	Extended	X	2	7	
	Disk	X	2	7	
TRS Color	Level I				
	Extended	X	1	6	
	Disk	X	1	6	
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

*Notes*

1. A comma must be used between the line numbers, not a hyphen.
2. If line-number-2 is not an existing line in the program, the next higher line in the program is used.
3. If line-number-1 is 0, only line 0 is deleted, regardless of the value of line-number-2.
4. The statement must have two line numbers; to delete a single line use DEL 100,100.
5. If DEL is used, it will list as DELETE.
6. If only line-number-1 and the hyphen are specified, all lines from the specified one to the end of the program are deleted. The command DEL — deletes the entire program.
7. A period can be used to indicate the current line.

**delimiter**

A delimiter is a character, or group of characters, that is used to set off, or delimit, a certain group of data.

A delimiter cannot itself appear in the material it delimits unless some special convention is used. For example, in BASIC the delimiter for string constants is the double-quotation character, so a double-quotation character cannot appear in a string constant.

**DIM***Format*

DIM {variable-name (value [,value] ... ),} ...

The DIM statement specifies the dimensions of an array. It overrides the default value of 10 for each dimension of an array. Arrays can be either numeric or string, as determined by the variable-name. The default for an array is eleven elements. An array cannot be redimensioned in a program without first being erased.

In general, the size of arrays is limited by the amount of storage available.

*Example*

The statement DIM X(2,3,4), Y\$(15) defines X as a three-dimensional array with three elements in the first dimension, four in the second, and five in the third; it also defines Y\$ as a string array of 16 elements.

**See:** *array, CLEAR, ERASE, OPTION BASE, subscript.*

DIM
-----

System		In	Format	Notes	Alternate Commands
APPLE	Integer	X		1, 6	
	Applesoft	X		2	
	DOS	X		2	
	Microsoft	X		5	
IBM	Cassette	X		5	
	Disk	X		5	
	Advanced	X		5	
TRS Mod III	Level I				
	Extended	X		3	
	Disk	X		3	
TRS Color	Level I	X		4	
	Extended	X			
	Disk	X			
Commodore	VIC 20	X			
ATARI	400/800	X		6, 7	
ANSI	Minimum	X			

*Notes*

1. Only one-dimensional numeric arrays are allowed.
2. Array values are cleared by RUN.
3. DIM must be used for an array of more than three dimensions.

4. A 4K system can support only one-dimensional arrays.
5. The elements of the array are set to zero or null by DIM.
6. When used with a string variable, DIM specifies the number of characters in the variable. String variables must be dimensioned before they are used; there is no default. Array values are not set to zero by RUN, CLR, or RESET.
7. Only one- and two-dimensional numeric arrays are allowed.

**DIR***Format*

DIR drive-number

DIR displays the directory of the specified drive. If no drive is specified, drive 0, or the one specified as the default by the DRIVE statement, is used.

The drive-number must be an integer between 0 and 3.

*See: CATALOG, FILES.*

DIR
-----

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				CATALOG
	DOS				CATALOG
	Microsoft				FILES
IBM	Cassette				
	Disk				FILES
	Advanced				FILES
TRS Mod III	Level I				
	Extended				
	Disk	X			
TRS Color	Level I				
	Extended				
	Disk	X			
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

**DISKINI***Format*

DISKINI drive-number

The DISKINI command formats the disk in the specified drive. The previous contents of the disk and the contents of memory are erased.

The drive-number must be an integer between 0 and 3, inclusive.

## DISKINI

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft				
IBM	Cassette				
	Disk				
	Advanced				
TRS Mod III	Level I				
	Extended				
	Disk	X			
TRS Color	Level I				
	Extended				
	Disk	X		1	
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

*Note*

1. Must be spelled DSKINI.

**disks**

All systems use a 5.25-inch (13.33-centimeter) single-sided, double-density mini-floppy diskette. The following table lists some of the properties of the disks that are implementor-defined.

Specification	IBM	TRS Color	TRS Mod III	Apple
Number of tracks	40	35	40	35 (1)
Sectors per track	8	18	18	16 (2)
Bytes per sector	512	256	256	256
Bytes per track	4096	4608	4608	4096
Formatted capacity (kilobytes)	163.8 (4)	179	175	143 (3)
Directory is on track number:	0	17	17	17
Space is allocated:	One sector at a time	Units of 2304 bytes	Units of 768 bytes	One sector at a time

*Notes*

1. 31 tracks for the user.
2. Older versions of Apple had only 13 sectors per track. There is a utility program called Muffin, to convert from 13- to 16-track format.
3. 127K bytes for the user.
4. Double-density disks with a formatted capacity of 320K bytes are supported by DOS 1.1.

**DLOAD***Format*

DLOAD program-name, baud-rate

The DLOAD command loads the specified machine language program at the rate specified. It is used for down-line loading from another computer.

The baud-rate must be either 0 or 1; 0 is 300 baud, 1 is 1200 baud.

DLOAD is supported only by TRS Color Basic (Extended and Disk).

*See: CALL, CLOADM, EXEC, SPEED, SYS, USR.*

**DOS***Format*

DOS

The DOS command transfers control to the disk operating system and displays the DOS menu. If DOS has not been booted, the computer goes to the Memo Pad mode.

DOS is supported only by ATARI BASIC.

*See: SYSTEM.*

**double-  
precision  
number**

*See: real number.*

**DRAW***Format 1*

DRAW arithmetic-expression [AT x-value, y-value] ...

*Format 2*

DRAW command-string

*Format 1*

This form of the DRAW statement draws a "shape." A shape is defined by a series of commands called a shape description, which is located in a shape table. The shape selected is the one whose ordinal position in the shape table matches the value of the expression.

If the AT clause is specified, the drawing of the shape begins at the specified point. If this clause is not specified, the shape starts at the last point plotted by a HLOT, DRAW, or XDRAW statement.

The value of the expression must be between 0 and 255 and must not exceed the number of shape definitions in the table. The x-value must be between 0 and 279; the y-value must be between 0 and 191. (All ranges, inclusive.)

*Format 2*

This form of the DRAW statement draws an object that is defined by the command-string, which can be a literal or in a variable.

The various commands that can be included in this string are shown below. They can be separated by semicolons, but this is not mandatory.

Command	Meaning
<i>Uvalue</i>	Draw up (0 degrees)
<i>Rvalue</i>	Draw right (90 degrees)
<i>Dvalue</i>	Draw down (180 degrees)
<i>Lvalue</i>	Draw left (270 degrees)
<i>Evalue</i>	Draw at 45 degrees
<i>Fvalue</i>	Draw at 135 degrees
<i>Gvalue</i>	Draw at 225 degrees
<i>Hvalue</i>	Draw at 315 degrees

(Degrees are measured clockwise.)

For IBM, the distance moved is the value times the scaling factor. The value can also be specified by a variable; if so, it must be written as

E = variable;

In this case, the use of the semicolon is mandatory.

For TRS Color, the distance is the number of points specified by the value. If no value is specified, 1 is used. Values must be integer constants; nonintegers are truncated.

Following is a list of the commands and their meanings.

### **B**

Move, but do not draw the line.

### **N**

Move, but return to the starting point when done.

### **Avalue**

Sets the angular rotation. The value must be an integer from 0 to 3, with meanings as follows:

- 0    No rotation
- 1    90 degrees clockwise
- 2    180 degrees clockwise
- 3    270 degrees clockwise

For IBM, figures rotated 90 or 270 degrees are scaled so that they keep the same perspective as if plotted at 0 or 270 degrees.

### **Svalue**

Scale factor. For IBM the value must be between 1 and 255; this value divided by 4 is the scale factor. For TRS Color, the value must be between 1 and 62; it represents the scale in  $\frac{1}{4}$  increments. The default is S4, which is a 1 : 1 scale.

### **Cvalue**

Specifies the color. For IBM, in medium resolution this must be between 0 and 3; for high resolution it must be 0 or 1. For TRS Color, it must be between 0 and 8; if omitted, the foreground color is used.

### **Xstring-variable;**

Executes the specified string as if it were in the command. The trailing semicolon is required.

### **M[+ | -] x-value, y-value**

Moves the draw position to the specified coordinate. If a plus or minus precedes the x-value, the move is relative to the current position. If no plus or minus precedes the x-value, the move is to the absolute location specified by the coordinates. Unless the M is preceded by a B, an unwanted line is usually plotted.

For IBM the *x*-value must be between 0 and 319 (medium resolution) and 0 and 639 (high resolution); the *y*-value must be between 0 and 199. For TRS Color, the *x*-value must be between 0 and 255, the *y*-value between 0 and 191.

*Note:* In TRS if PMODE is 0 or 1 and E, F, G, or H is used, then if the length is odd and at least one of the coordinates is odd, lines drawn by F and H have a glitch at the midpoint; if both coordinates are even, lines drawn by E and G have the glitch.

*See: CIRCLE, HLIN, HPLOT, PAINT, PMODE, ROT, SCALE, shape, VARPTR\$, VLIN, XDRAW.*

DRAW
------

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft	X	1		
	DOS	X	1		
	Microsoft				
IBM	Cassette				
	Disk				
	Advanced	X	2		
TRS Mod III	Level I				
	Extended				
	Disk				
TRS Color	Level I				
	Extended	X	2		
	Disk	X	2		
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

## DRAWTO

*Format*

DRAWTO *x*-coordinate, *y*-coordinate

The DRAWTO statement draws a line from the last point displayed by a PLOT to the specified location. Either coordinate can be an arithmetic-expression. The color of the line is the same as the one used for PLOT.

DRAWTO is supported only by ATARI BASIC.

*See: COLOR, HPLOT, PLOT.*

**DRIVE***Format*

DRIVE drive-number

The DRIVE command changes the default drive to the one specified. The drive-number must be between 0 and 3.

DRIVE is supported only by TRS Color Disk BASIC.

**DSKI\$***Format*

DSKI\$ drive, track, sector, string-variable-1, string-variable-2

The DSKI\$ statement performs a direct disk read. It ignores the directory and other indexing information and reads 256 bytes from a specified location on the disk. The first 128 bytes go into string-variable-1; the second 128 bytes go into string-variable-2.

Drive must be an integer between 0 and 3.

Track must be an integer between 0 and 34.

Sector must be an integer between 1 and 18.

DSKI\$ is supported only by TRS Color Disk.

*See: DSKO\$, NOTE, POINT.*

**DSKINI***See: DISKINI***DSKO\$***Format*

DSKO\$ drive, track, sector, string-variable-1, string-variable-2

The DSKO\$ statement performs a direct disk write. It ignores the directory and other indexing information and writes 256 bytes of data from two variables to a specified area on the disk. The first 128 bytes are taken from string-variable-1; the second 128 bytes are taken from string-variable-2.

Drive must be an integer between 0 and 3.

Track must be an integer between 0 and 34.

Sector must be an integer between 1 and 18.

DSKO\$ is supported only by TRS Color Disk BASIC.

*See: DSKI\$, NOTE, POINT.*

**DSP***Format*

DSP variable-name

The DSP command enables tracing of a particular variable. Every time the variable is changed, the screen shows the line number, name, and value.

*Example*

```

100 DSP X1
120 DSP X2
140 X1 = 2 : X2 = 2 : X3 = 2
160 X1 = 3 : X2 = 3 : X3 = 2
180 X1 = X2 + X3
200 X3 = X3 + X2
220 END

```

**Output**

```

140 X1 = 2
140 X2 = 2
160 X1 = 3
160 X2 = 3
180 X1 = 5

```

DSP is supported only by Apple Integer BASIC.

*See: TRACE, TRON.*

**dummy variable**

1. A dummy variable is a variable that is specified in the definition of a function but which has no effect on the function's action. It simply holds a place for the argument that will be specified when the function is used in the program. In the statement

$$\text{DEF FNC (K) = (K * K * K)}$$

the variable K is a dummy variable. When the function is invoked, as, for example, by  $Z = \text{FNC}(X)$ , the value of X determines the value of the function; the value of K is irrelevant.

2. A dummy variable is an argument given to a function that does not affect the value that the function returns, but is necessary to meet syntactic requirements. When one executes  $Y = \text{POS}(X)$ , the value of X does not affect the value returned; therefore, X is a dummy variable. It would perhaps be more appropriate to call X a "dummy argument," but current usage dictates the former term.

*See: argument, DEF FN.*

**EDIT***Format*

EDIT line-number

The EDIT statement displays the specified line and awaits editing commands. (For details of the individual commands, see the appropriate reference manual.)

EDIT

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft	X		1,2	
IBM	Cassette	X		1	
	Disk	X		1	
	Advanced	X		1	
TRS Mod III	Level I				
	Extended	X		1	
	Disk	X		1	
TRS Color	Level I				
	Extended	X			
	Disk	X			
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

*Notes*

1. A period (.) can be used to represent the current line.
2. After typing the line number, the system waits for an editing sub-command.

**ELSE***See: IF.***END***Format*

END

The END statement marks the end of the program. It can also be used to separate the main program from a subroutine. When the END statement is encountered, control returns to the command level. A “BREAK AT” message is not displayed.

*See: CONT, STOP.*

END
-----

System		In	Format	Notes	Alternate Commands
APPLE	Integer	X		1	
	Applesoft	X		2	
	DOS	X		2	
	Microsoft	X		2, 3	
IBM	Cassette	X		2	
	Disk	X		2, 3	
	Advanced	X		2, 3	
TRS Mod III	Level I	X		1	
	Extended	X		2	
	Disk	X		2	
TRS Color	Level I	X		2	
	Extended	X		2	
	Disk	X		2	
Commodore	VIC 20	X		2	
ATARI	400/800	X		2, 4	
ANSI	Minimum	X			

### Notes

1. Required.
2. Optional.
3. Closes all disk files.
4. Turns off sounds.

## ENTER

### Format

ENTER “file-specification”

The ENTER command loads the specified program from tape without clearing the old program. The program must have been stored using LIST. This command can also be used with disk files. ENTER is usually used in direct mode.

ENTER is supported only by ATARI BASIC.

*See: CLOAD, LIST, LOAD.*

## entry point

The term “entry point” is usually applied to a machine language routine or subroutine. It is the address where execution of the routine begins. It is also called the “transfer address.”

## EOF

*Format*

EOF (file-number)

The EOF function returns a false (0) if data exist in the file and a true (–1) if the end of file has been reached. The file being tested must be sequential.

This is a good way to tell if a file exists. Immediately after opening the file, check EOF; if it is true, the file did not exist.

*Example*

```
100 OPEN "MYFILE.DAT" AS #1
120 IF EOF(1) NOT = 0 THEN (file did not exist)
```

EOF

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft	X			
IBM	Cassette	X		1	
	Disk	X		1	
	Advanced	X		1	
TRS Mod III	Level I				
	Extended				
	Disk	X			
TRS Color	Level I	X		2	
	Extended	X		2	
	Disk	X		2	
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

*Notes*

1. For a communications buffer,  $-1$  means that the buffer is empty. The file-number must be between 1 and the maximum number of files allowed.
2. For cassette files, the file-number is  $-1$ ; the keyboard is 0.

**EQV***Format*

argument-1 EQV argument-2

The equivalence function, EQV, is a logical function of two arguments. It has a value of true if both its operands have the same truth value, and a value of false otherwise.

The arguments can be relations, logical variables, or anything that can be evaluated as true or false. This is the inverse of the exclusive or (XOR) function. In formal logic the equivalence function is "P if and only if Q."

Truth Table for EQV

p	q	p EQV q
F	F	T
F	T	F
T	F	F
T	T	T

If EQV is not implemented, it can be calculated by

DEF FNEQV (P,Q) = (P AND Q) OR (NOT (P OR Q))

*Example*

100 IF A EQV B THEN GOTO 200

If A and B are either both true or both false, control is transferred to line 200.

*See: logical functions, XOR.*

EQV

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft	X			
IBM	Cassette	X			
	Disk	X			
	Advanced	X			
TRS Mod III	Level I				
	Extended				
	Disk				
TRS Color	Level I				
	Extended				
	Disk				
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

**ERASE***Format*

ERASE {array-name}...

The ERASE statement eliminates the specified arrays. Once an array has been erased, it can be redimensioned.

*See: array, DIM, OPTION BASE, subscript.*

ERASE

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft	X			
IBM	Cassette	X			
	Disk	X			
	Advanced	X			
TRS Mod III	Level I				
	Extended				
	Disk				
TRS Color	Level I				
	Extended				
	Disk				
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

ERL

Format

ERL

The ERL function has as its value the line number in which the most recent error occurred. If no error has occurred, ERL has a value of 0.

Example

```
100 ON ERROR GO TO 1000
120 GET #1
140 GET #2
    *
    *
    *
1000 IF ERL = 120 ... (error occurred in file 1)
1020 IF ERL = 140 ... (error occurred in file 2)
```

See: ERR, ERROR, ON ERROR.

ERL
-----

System		In	Format	Alternate Commands
APPLE	Integer			
	Applesoft			PEEK(218)+PEEK(219)*256
	DOS			PEEK(218)+PEEK(219)*256
	Microsoft	X		
IBM	Cassette	X		
	Disk	X		
	Advanced	X		
TRS Mod III	Level I			
	Extended	X		
	Disk	X		
TRS Color	Level I			
	Extended			
	Disk			
Commodore	VIC 20			
ATARI	400/800			PEEK(186)+PEEK(187)*256
ANSI	Minimum			

ERR

Format

ERR

The ERR function has as its value a code that indicates the most recent

error that has occurred. For the meaning of the codes for each system, see "error codes." If no error has occurred, ERR has a value of zero.

*Example*

```

100 ON ERROR GO TO 1000
120 GET # 1
    *
    *
    *

1000 IF ERR = 57 THEN ...    (process for error 57)
1020 IF ERR = 62 THEN ...    (process for error 62)

```

*See: ERL, ERROR, ON ERROR.*

ERR
-----

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				PEEK(222)
	DOS				
	Microsoft	X			
IBM	Cassette	X			
	Disk	X			
	Advanced	X			
TRS Mod III	Level I				
	Extended				
	Disk	X		1	
TRS Color	Level I				
	Extended				
	Disk				
Commodore	VIC 20				
ATARI	400/800				PEEK(195)
ANSI	Minimum				

*Note*

1. The true value is given by  $ERR/2 + 1$ .

## ERROR

*Format*

ERROR arithmetic-expression

The ERROR statement has two distinct uses: to simulate the occurrence of an error, and to specify a user-defined error code.

If the statement **ERROR 200**, is executed, and an error 200 is already defined, control proceeds as if error 200 had actually occurred. This facilitates testing error-handling routines. If error 200 is not defined, the statement defines it; **ERR** is set to the value 200 and any **ON ERROR** statement is then executed.

*Example*

```

100 ON ERROR GO TO 1000
120 ERROR 200
      *
      *
      *
1000 IF ERR = 200 THEN ...    (process for error)

```

*Example*

```

100 ERROR 22
RUN

```

**Output**

MISSING OPERAND IN 100

*See: **ERL**, **ERR**, **ON ERROR**.*

**ERROR**

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft	X		1	
IBM	Cassette	X		1	
	Disk	X		1	
	Advanced	X		1	
TRS Mod III	Level I				
	Extended	X		2	
	Disk	X		2	
TRS Color	Level I				
	Extended				
	Disk				
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

*Notes*

1. The value of the expression can be between 0 and 255.
2. The value of the expression can be between 1 and 23; disk errors cannot be simulated. This cannot be used for user-defined error codes.

**errors**

Following are lists of the errors that are associated with the various systems. If appropriate, the error code (which is the value that ERR is set to) is also included. For details as to the etiology of the various errors, refer to the appropriate reference manual.

**Apple**

```
.> 255 ERR
.> 32767 ERR
16 FORS ERR
16 GOSUBS ERR
BAD BRANCH ERR
BAD NEXT ERR
BAD RETURN ERR
DIM ERR
MEM FULL ERR
NO END ERR
RANGE ERR
RETYPE ERR
STOPPED AT line-number
STR OVFL ERR
STRING ERR
SYNTAX ERR
TOO LONG ERR
```

**Applesoft**

Messages are printed in the form

```
?“xx” ERROR IN line-number
```

where “xx” is one of the following messages.

```
CAN'T CONTINUE
0 NEXT WITHOUT FOR
16 SYNTAX ERROR
22 RETURN WITHOUT GOSUB
42 OUT OF DATA
53 ILLEGAL QUANTITY
```

- 69 OVERFLOW
- 77 OUT OF MEMORY
- 90 UNDEF'D STATEMENT
- 107 BAD SUBSCRIPT
- 120 REDIM'D ARRAY
- 133 DIVISION BY ZERO
- 163 TYPE MISMATCH
- 176 STRING TOO LONG
- 191 FORMULA TOO COMPLEX
- 224 UNDEF'D FUNCTION
- 254 ILLEGAL DIRECT
- 255 CTL-C INTERRUPT ATTEMPTED

#### **Applesoft Disk Operating System**

- 1 LANGUAGE NOT AVAILABLE
- 2 RANGE ERROR
- 3 RANGE ERROR
- 4 WRITE PROTECTED
- 5 END OF DATA
- 6 FILE NOT FOUND
- 7 VOLUME MISMATCH
- 8 I/O ERROR
- 9 DISK FULL
- 10 FILE LOCKED
- 11 SYNTAX ERROR
- 12 NO BUFFERS AVAILABLE
- 13 FILE TYPE MISMATCH
- 14 PROGRAM TOO LARGE
- 15 NOT DIRECT COMMAND

#### **IBM and Microsoft**

(Unless otherwise specified, these errors apply to both systems.)

- 1 NEXT without FOR
- 2 Syntax error
- 3 RETURN without GOSUB
- 4 Out of data
- 5 Illegal function call
- 6 Overflow
- 7 Out of memory
- 8 Undefined line number
- 9 Subscript out of range
- 10. Duplicate definition of array
- 11 Attempted division by zero
- 12 Illegal direct mode command

- 13 Type mismatch
- 14 Out of string space
- 15 String too long
- 16 String formula too complex
- 17 Illegal use of CONT
- 18 Undefined USR function
- 19 No RESUME
- 20 RESUME without error
- 21 Unprintable error (Microsoft only)
- 22 Missing operand
- 23 Line buffer overflow
- 24 Device timeout (IBM only)
- 25 Device fault (IBM only)
- 26 FOR without NEXT
- 27 Out of paper (IBM only)
- 29 WHILE without WEND
- 30 WEND without WHILE
- 31 Reset error (Microsoft only)
- 32 Illegal graphics statement (Microsoft only)
  
- 50 FIELD overflow
- 51 Internal error
- 52 Bad file number
- 53 File not found
- 54 Bad file mode
- 55 File already open
- 57 Device I/O error
- 58 File already exists
  
- 61 Disk full
- 62 Input past end of file
- 63 Bad record number
- 64 Bad file name
- 66 Direct statement is in file
- 67 Too many files
- 68 Device unavailable (IBM only)
  - Disk read only (Microsoft only)
- 69 Communication buffer overflow (IBM only)
  - Drive select error (Microsoft only)
- 70 Disk write protect (IBM only)
  - File read only (Microsoft only)
- 71 Disk not ready (IBM only)
- 72 Disk media error (IBM only)
- 73 Use of advanced BASIC in disk mode (IBM only)
  - Unprintable error (Microsoft only)

**TRS Mod III**

- 1 NF NEXT without FOR
- 2 SN Syntax error
- 3 RG RETURN without GOSUB
- 4 OD Out of data
- 5 FC Illegal function call
- 6 OV Overflow
- 7 OM Out of memory
- 8 UL Undefined line number
- 9 BS Subscript out of range
- 10 DD Duplicate definition of array
- 11 /0 Attempted division by zero
- 12 ID Illegal direct mode command
- 13 TM Type mismatch
- 14 OS Out of string space
- 15 LS String too long
- 16 ST String formula too complex
- 17 CN Illegal use of CONT
- 18 NR No RESUME
- 19 RW RESUME without error
- 20 UE Unprintable error
- 21 MO Missing operand
- 22 FD Bad file data
- 23 L3 Disk BASIC only

**TRS Mod III Disk Operating System**

- 51 Field overflow
- 52 Internal error
- 53 Bad file number
- 54 File not found
- 55 Bad file mode
- 58 Disk I/O error
- 62 Disk full
- 63 Input past EOF
- 64 Bad record name
- 65 Bad file name
- 67 Direct statement in file
- 68 Too many files
- 69 Disk write protect
- 70 File access violation

**TRS Color**

- /0 Division by zero attempted
- A0 File already open
- BS Bad subscript

CN Can't continue  
DD Redimensioned array  
DN Device number error  
DS Direct statement is in file  
FC Illegal function call  
FD Bad file data  
FM Bad file mode  
ID Illegal direct statement  
IE Input past end of file  
IO Input/Output error  
LS String too long  
NF NEXT without FOR  
NO File not open  
OD Out of data  
OM Out of memory  
OS Out of string space  
OV Overflow  
RG RETURN without GOSUB  
SN Syntax error  
ST String formula too complex  
TM Type mismatch  
UL Undefined line

### **TRS Color Disk Operating System**

AE File already exists  
BR Bad record number  
DF Disk full  
ER Past end of record (direct access)  
FN Bad file name  
FO Field overflow  
FS Bad file structure  
NE Can't find file  
OB Out of buffer space  
SE SET to a non-FIELDed string  
VF Verification  
WP Write protected

### **Commodore VIC-20**

BAD DATA  
BAD SUBSCRIPT  
CAN'T CONTINUE  
DEVICE NOT PRESENT  
DIVISION BY ZERO  
EXTRA IGNORED  
FILE NOT FOUND

FILE NOT OPEN  
FILE OPEN  
FORMULA TOO COMPLEX  
ILLEGAL DIRECT  
ILLEGAL QUANTITY  
LOAD  
NEXT WITHOUT FOR  
NOT INPUT FILE  
NOT OUTPUT FILE  
OUT OF DATA  
OUT OF MEMORY  
OVERFLOW  
REDIM'D ARRAY  
REDO FROM START  
RETURN WITHOUT GOSUB  
STRING TOO LONG  
SYNTAX  
TYPE MISMATCH  
UNDEF'D FUNCTION  
UNDEF'D STATEMENT  
VERIFY

#### ATARI

2 Memory insufficient  
3 Value error  
4 Too many variables  
5 String length error  
6 Out of data error  
7 Number greater than 32767  
8 Input statement error  
9 Array of string DIM error  
10 Argument stack overflow  
11 Floating-point overflow/underflow error  
12 Line not found  
13 No matching FOR statement  
14 Line too long error  
15 GOSUB or FOR line deleted  
16 RETURN error  
17 Garbage error  
18 Invalid string character  
19 LOAD program too long  
20 Device number larger  
21 LOAD file error  
  
128 BREAK abort  
129 IOCB (Input/Output Control Block)

130 Nonexistent device  
131 IOCB write only  
132 Invalid command  
133 Device or file not open  
134 BAD IOCB number  
135 IOCB read-only error  
136 EOF (end of file)  
137 Truncated record  
138 Device timeout  
139 Device NAK (Negative AcKnowledgegment)  
140 Serial bus  
141 Cursor out of range  
142 Serial bus data frame overrun  
143 Serial bus data frame checksum error  
144 Device done error  
145 Read after write compare error  
146 Function not implemented  
147 Insufficient RAM  
160 Drive number error  
161 Too many OPEN files  
162 Disk full  
163 Unrecoverable system data I/O error  
164 File number mismatch  
165 File name error  
166 POINT data length error  
167 File locked  
168 Command invalid  
169 Directory full  
170 File not found  
171 POINT invalid

**EXEC***Format 1*

EXEC [address]

*Format 2*

EXEC file-name [,Rpointer] [,Ddevice] [,Sslot] [,Vvolume]

*Format 1*

This form of the EXEC command transfers control to a machine language program at the specified address. If no address is specified, control is transferred to the address specified in the last CLOADM that was executed.

*Format 2*

This form of EXEC operates similarly to RUN; the file specified must be a text file that contains BASIC and DOS commands in the same form as they would be issued from the keyboard. The file is then executed as if these commands were typed in. This is a way of performing command file execution.

Only one EXEC command can be in effect at any given time; if the file being executed itself contains an EXEC command, the first file is closed and the file specified in the new EXEC command is opened and executed.

If a file has been opened by an EXEC, a CLOSE will have no effect on it. Only when the file has been completely executed will it stop and be closed.

If the file being executed contains a RUN command, the designated program runs and, when completed, execution of the original file continues.

If a pointer value is specified, the pointer is set to the position specified and execution begins at this point in the file. The position is calculated from the start of the file, not the current position.

This command must be executed, preceded by a CONTROL-D, as part of a PRINT statement:

```
PRINT CHR$(4); "EXEC MYFILE, R1000"
```

The allowable values for parameters and their defaults are shown below. Once a value has been changed, the last value specified is the default.

Parameter	Range	Default
pointer	0 to 65535	None
drive	1 or 2	Drive 1
slot	1 to 7	Slot from which DOS was booted
volume	1 to 254	Volume from which DOS was booted

*See: CALL, CLOADM, DLOAD, SYS, USR.*

EXEC

System		In	Format	Notes	Alternate Commands
APPLE	Integer				CALL
	Applesoft				CALL
	DOS	X	2		
	Microsoft				CALL
IBM	Cassette				CALL
	Disk				CALL
	Advanced				CALL
TRS Mod III	Level I				
	Extended				
	Disk				
TRS Color	Level I				
	Extended	X	1	1	
	Disk	X	1	1	
Commodore	VIC 20				SYS
ATARI	400/800				
ANSI	Minimum				

*Note*

1. An address is required.

## EXP

*Format*

EXP (arithmetic-expression)

The exponential function, EXP, has as its value the constant  $e$  (2.71828183), raised to the power indicated by the expression. This is the inverse function of LOG.

If  $Y = \text{EXP}(X)$ , then  $Y = e^X$ .

*Example*

```
100 Y = EXP(13)
120 PRINT EXP(1), Y, EXP(85.5312)
```

**Output**

2.718282    .4424134E06    1.398705E37

(As the value of  $X$  increases beyond 87, the possibility of overflow increases.)

*See: CLOG, LOG.*

EXP
-----

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft	X			
	DOS	X			
	Microsoft	X		2	
IBM	Cassette	X		1	
	Disk	X		1	
	Advanced	X		1	
TRS Mod III	Level I				
	Extended	X		2	
	Disk	X		2	
TRS Color	Level I				
	Extended	X		2	
	Disk	X		2	
Commodore	VIC 20	X		1	
ATARI	400/800	X			
ANSI	Minimum	X			

*Notes*

1. The expression must not exceed 88.02969.
2. The expression must not exceed 87.3365.

**exponen-  
tiation**

Exponentiation is the raising of a number to a power.

The symbol used to denote exponentiation varies among implementations, as shown in the following chart:

Symbol	Implementation
(uparrow) ↑	Commodore, Microsoft, TRS Color
(carat) ^	Apple, Applesoft, ATARI, IBM
(uparrow) ↑ or (left bracket) [	TRS Mod III

*Note:* The value of a negative number raised to an even power, written  $-X^K$ , will be positive or negative according to the hierarchy of operations.

*See: arithmetic operations.*

**expression**    *See: arithmetic expression, concatenation, string expression.*

# F

## FIELD

*Format*

FIELD #file-number, {arithmetic-expression AS field-name,...}

The FIELD command organizes the space in a buffer into named entities called fields. The file to be organized must be open, and all data must be in string form. A buffer must be fielded before data can be accessed with a GET or PUT statement. The size and name of each field must be specified; the arithmetic-expression defines the length of the field; the field-name, its name. A buffer can have multiple field definitions for the same data. The sum of all the fields' lengths must not be greater than the record length; in practice, it should equal the record length.

The field-name must conform to the rules for a string variable name. However, these fields are not string variables and they do not occupy string space. If a field-name is used in an assignment statement to the left of the equal sign, it will no longer access the field in the buffer.

*See: CVD, CVI, CVN, CVS, GET, MKD\$, MKI\$, MKN\$, MKS\$, OPEN, PUT.*

### FIELD

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft	X		1	
IBM	Cassette				
	Disk	X		1, 2	
	Advanced	X		1, 2	
TRS Mod III	Level I				
	Extended				
	Disk	X		3	
TRS Color	Level I				
	Extended				
	Disk	X			
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

*Notes*

1. The number sign (#) is optional.
2. The file-number can range from 1 to the maximum number of files currently allowed.
3. A buffer-number must be used instead of a file-number, it must be between 1 and 15 inclusive. The number sign (#) cannot be used.

**file specification**

Each system has a different convention for naming files, as shown in the following summary.

*TRS Color*

name [. extension] [: drive]

The name can have from one to eight characters; it cannot contain a colon or a zero, but embedded spaces are allowed. The extension can have from zero to three characters. Either a period (.) or a stroke (/) must separate the name from the extension. The drive must be from 0 to 3; if not specified, drive 0, or the default drive, is used.

*TRS Mod III*

name [/extension] [. password] [: drive]

The name can have from one to eight characters; the first character must be a letter, remaining characters can be letters or numbers. Embedded spaces are not allowed. The extension can have from zero to three characters, the first character must be a letter. The password can have from one to eight characters. The drive must be from 0 to 3. If no drive is specified, the system begins with drive 0 and searches for the file; if it is not found, the search continues on drive 1; and so on.

*Apple*

file-name

The file-name can have from 1 to 30 characters. The first character must be a letter. The comma, CONTROL-M, and Carriage Return are not permitted in the name. Leading spaces are ignored, but the name can contain embedded spaces.

The equal sign (=) can be used as a wild card.

*IBM*

[device:] filename [. extension]

The device can be any of the following:

A: B:	First and second disk drives (Disk and Advanced only)
CAS1:	Cassette tape player
COMx:	Asynchronous communications adapter; X can be 1 or 2 (Disk and Advanced only)
SCRN:	Screen
LPTx:	Line printer; X can be 1, 2, or 3; Cassette BASIC supports only one line printer
KYBD:	Keyboard

**Filename**

In Cassette BASIC, the filename can have up to eight characters. The colon, hex "00", and hex "FF" cannot appear in the name.

In Disk and Advanced BASIC, the filename can have from one to eight characters; the extension, zero to three characters. Embedded spaces are not allowed. The following characters can be used for the filename and extension.

A through Z (lowercase is converted to uppercase)	Parentheses: ) (
0 through 9	Ampersand (&)
Dollar sign (\$)	Left apostrophe ( ' )
Right apostrophe ( ' )	Hyphen (-)
Underscore ( _ )	Number sign ( # )
At sign ( @ )	Percent sign ( % )
Braces ( { and } )	Less-than sign ( < )
Greater-than sign ( > )	Tilde ( ~ )
Backslash ( \ )	Exclamation Point ( ! )
Carat ( ^ )	Vertical dashed line (   )

Any other character is invalid and functions as a delimiter, truncating the name.

The question mark (?) and asterisk (\*) can be used as wild cards.

*Microsoft*

[device:] filename [. extension]

The filename can have from one to eight characters; they must be upper-case, no conversion is done from lowercase. Embedded spaces are not allowed. The extension can be from zero to three characters.

### *ATARI*

device-code [device-number] : [filename] . [extension]

### **Device Codes**

C:     Cassette  
D:     Disk (up to four drives are supported)  
E:     Screen Editor  
K:     Keyboard  
P:     Printer  
R:     RS-232 Interface  
S:     Screen

The device number is optional. If not specified, number 1 is the default.

The filename can have from zero to eight characters; the extension, zero to three. (Filenames are not used with cassette files.)

The question mark (?) and asterisk (\*) can be used as wild cards.

***See: wild card.***

## **FILES**

### *Format 1*

FILES [file-specification]

### *Format 2*

FILES buffer-number, buffer-size

### *Format 1*

This form of the FILES command displays the names of files currently on the specified disk. If no file is specified, all files are listed. If no drive is specified, the current drive is used.

An asterisk (\*) as the first character of the file-name or extension indicates a wild card. That is, any file or extension is considered to match the asterisk. A question mark (?) in the file name or extension will cause any character in *that* position to match.

*Format 2*

This form of FILES defines the number of disk buffers and the total number of bytes for the buffers.

The buffer-number specifies the number of buffers allocated. It must have a value between 1 and 15, the default is 2. The buffer-size specifies the *total* number of bytes allocated for all the buffers; the default is 256 bytes.

*See: CATALOG, DIR.*

## FILES

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				CATALOG
	DOS				
	Microsoft	X	1		
IBM	Cassette				
	Disk	X	1		
	Advanced	X	1		
TRS Mod III	Level I				
	Extended				
	Disk				DIR
TRS Color	Level I				
	Extended				
	Disk	X	2		
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

**firmware**

Firmware is another term for read-only memory (q.v.).

**FIX***Format*

FIX (arithmetic-expression)

The FIX function has as its value the integer part of the argument. The fractional part, if any, is truncated. FIX differs from INT in that for a negative number it does not return a value less than the number.

*Example*

100 Y = FIX(X)

**Output for Different Values of X Compared with INT**

X	FIX(X)	INT(X)
-2.9	-2	-3
-2.1	-2	-3
-2.0	-2	-2
+2.1	2	2
+2.9	2	2

*See: CINT, INT.*

FIX
-----

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft	X			
IBM	Cassette	X			
	Disk	X			
	Advanced	X			
TRS Mod III	Level I				
	Extended	X			
	Disk	X			
TRS Color	Level I				
	Extended	X			
	Disk	X			
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

**FLASH**

*Format*

**FLASH**

The FLASH command sets the video mode such that subsequent output alternates between foreground color on background and background color on foreground. It does not affect the characters currently on the screen when the command is executed, nor characters typed into the computer.

FLASH is supported only in Applesoft and Apple DOS.

*See: INVERSE, NORMAL.*

**floating-point number**      *See: real number.*

**floppy disks**      *See: disks.*

**FOR**      *Format*

FOR counter = starting-value TO ending-value [STEP increment]

The FOR statement works in conjunction with a NEXT statement that specifies the same counter. All statements between the FOR and this NEXT are the *range* of the FOR statement.

The FOR statement operates as follows:

1. The counter is set to the starting-value.
2. All statements in the range are executed.
3. When control reaches the NEXT statement, the specified increment is added to the counter.
4. The new value of the counter is then compared with the ending-value. If it does not exceed the ending-value, control returns to the first statement of the range and the entire range is executed again.
5. Action continues as above until the value of the counter exceeds that of the ending-value; at this point control passes to the statement following the NEXT.

All parameters must be numeric. Once the FOR statement has begun executing, changing the starting or ending value has no effect on its operation. However, changing the value of the counter *will* affect the operation.

The increment can be positive or negative. If negative, the ending-value should be less than the starting-value, and execution stops when the starting-value is less than the ending-value. If no value is specified, +1 is used.

*Example*

```
100 FOR I = 1 TO 10 STEP 2
120 PRINT I,
140 NEXT I
```

**Output**

1      3      5      7      9

*See: loop, NEXT, WHILE.*

FOR
-----

System	In	Format	Notes	Alternate Commands
APPLE	Integer	X	1	
	Applesoft	X	2	
	DOS	X	2	
	Microsoft	X	3	
IBM	Cassette	X	3	
	Disk	X	3	
	Advanced	X	3	
TRS Mod III	Level I	X		
	Extended	X		
	Disk	X		
TRS Color	Level I	X		
	Extended	X		
	Disk	X		
Commodore	VIC 20	X		
ATARI	400/800	X		
ANSI	Minimum	X		

*Notes*

1. Statements can be nested 16 deep
2. Statements can be nested 10 deep; the counter cannot be an integer.
3. If the increment is positive and the starting-value is initially greater than the ending-value, or if the increment is negative and the starting-value is initially less than the ending-value, the range is not performed. The counter cannot be double-precision.

**format**

In this book a format is a general description of a statement, function, or command. It indicates the specific ordering of the entry in a program. When an entry can appear in more than one way, numbered formats indicate the alternatives. In the formats the following conventions are used.

A word entirely in uppercase is a BASIC reserved word; as such it has a special meaning to BASIC and must be used where indicated and spelled exactly as shown. (Some BASICs do not even allow a reserved word to be embedded in a programmer-defined name.)

A word in lowercase indicates an item that is to be supplied by the programmer. It is usually a parameter or a variable.

Braces { } indicate mandatory entries; they are generally used to delimit the scope of an ellipsis.

Brackets [ ] indicate optional entries. This means that the material in the brackets can be included or not as the situation demands.

Ellipses (...) indicate that the preceding group can be repeated as often as necessary.

The vertical bar (|) separates alternatives. Only one of the alternatives can be chosen.

Any punctuation (comma, semicolon, parentheses) must be included as shown in the format unless the rules indicate otherwise.

Typical programmer-supplied items are:

- arithmetic-expression
- string-expression
- line-number
- file-specification

### *Example*

Consider the format for the DIM statement

`DIM {variable-name (value [,value]...) , } ...`

This is interpreted as follows. DIM is a reserved word and must be written where shown.

The two braces indicate that a variable-name, followed by a left parenthesis, a value, and a right parenthesis are the minimum that must be specified. Thus the simplest entry is

`DIM A$(10)`

The square brackets and ellipsis indicate that more than one value can appear in the parentheses, separated by commas. So another legal statement is

`DIM A$(10,10,5)`

The ellipsis after the braces indicates that more than one variable name can appear in the statement, thus:

`DIM A(5,5), B$(3,2,7), C(11)`

The format **MOTOR ON|OFF** indicates that the two valid forms of the statement are **MOTOR ON** and **MOTOR OFF**.

**FP**

*Format*

**FP**

The **FP** command invokes Applesoft BASIC.

**FP**

System		In	Format	Notes	Alternate Commands
APPLE	Integer	X			
	Applesoft	X		1	
	DOS	X		1	
	Microsoft				
IBM	Cassette				
	Disk				
	Advanced				
TRS Mod III	Level I				
	Extended				
	Disk				
TRS Color	Level I				
	Extended				
	Disk				
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

*Note*

1. Flushes the program currently in memory.

**FREE**

*Format 1*

**FREE (expression)**

*Format 2*

**FREE (drive-number)**

*Format 1*

This form of the **FREE** function has as its value the amount of storage currently available (in bytes). The expression is a dummy-variable, and can

be either arithmetic or string. However, since it is parsed, the expression must be valid.

Execution of FREE causes “housecleaning,” that is, the elimination of fragmented memory by the reordering of data. This operation can take from several seconds to minutes.

### *Format 2*

This form of FREE returns the number of free “granules” on the specified disk. (One granule = 2304 bytes.) The drive number can be 0 to 3.

*See: HIMEM:, LOMEM:, MEM.*

FREE
------

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft	X	1	1	
	DOS	X	1	1	
	Microsoft	X	1	2	
IBM	Cassette	X	1		
	Disk	X	1		
	Advanced	X	1		
TRS Mod III	Level I				
	Extended	X	1	3	
	Disk	X	1	3	
TRS Color	Level I				
	Extended				
	Disk	X	2		
Commodore	VIC 20				
ATARI	400/800	X	1	4	
ANSI	Minimum				

### *Notes*

1. If the value returned is negative, add 65536 to it to find the true number of free bytes.
2. Housekeeping is performed only when the argument is the null string.
3. If the expression is a string, the amount of string space available is returned. In this case it functions the same as MEM.
4. The dummy variable must be arithmetic.

**function**

A function is a predefined operation with zero, one, or several operands which, when invoked, returns a single value. A function can be either numeric or string; and can be user-defined or defined in BASIC.

*See: command, DEF FN, statement.*

# G

## GET

### *Format 1*

GET variable

### *Format 2*

GET (x-value-1, y-value-1) – (x-value-2, y-value-2), array-name [,G]

### *Format 3*

GET #buffer-number [,record-number]

### *Format 4*

GET #file-number, variable

### *Format 1*

This form of the GET statement inputs the last character typed at the keyboard. This character is not displayed on the screen, and a Carriage Return is not necessary to terminate the input.

### *Format 2*

This form of GET stores in an array the values of the points within a rectangular area of the screen. The first set of x and y values defines the upper left corner of the rectangle; the second set, the lower right corner. If G is specified, the points are stored with full graphics detail.

### *Format 3*

This form of GET reads a random access record and puts it into the designated buffer. The record that is read is specified by the number, which can be an expression. If no record-number is specified, the “next” record is read. The buffer-number must have a value between 1 and 15, inclusive.

*Format 4*

This form of GET moves one character (byte) from the file and puts it in the specified variable. It will pass Carriage Returns and special characters.

**See: COLOR, IN#, INKEYS, LOCATE, POSITION, PUT.**

GET
-----

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft	X	1	2, 3	
	DOS	X	1	2, 3	
	Microsoft	X	1, 3	1, 4	
IBM	Cassette				
	Disk	X	3	4	
	Advanced	X	2, 3	4, 5	
TRS Mod III	Level I				
	Extended				
	Disk	X	3	8	
TRS Color	Level I				
	Extended	X	2	9	
	Disk	X	3		
Commodore	VIC 20	X	1, 4	6, 7	
ATARI	400/800	X	4	10	
ANSI	Minimum				

*Notes*

1. A CONTROL-C returns a null; a left arrow or CONTROL-H may print as a null (Format 1).
2. A CONTROL-C will be accepted; it will not interrupt the program. If the variable is arithmetic, a nondigit returns zero.
3. If an ONERR ... GOTO and RESUME are used, two consecutive GET errors hang the system. If, instead of a RESUME, a GOTO is used, the forty-third GET error (not necessarily consecutive) causes a return to the monitor.
4. The record-number must be less than 32768; the number sign (#) is optional; and a file-number is used instead of a buffer-number.
5. Format 2 is valid only in graphics mode. It reads just the colors of the points. The G cannot be specified.
6. If a key is not pressed, the program does not wait but continues running.

7. One character is input from the device specified by buffer-number; if no number is specified, the keyboard is used but the character is not printed.
8. If an attempt is made to GET a fixed-length record beyond the end of file (EOF), a null record is returned, but no indication is given that this has occurred.
9. Uses a two-dimensional array to store the properties of the  $x$ -dimension in the first dimension of the array and the  $y$ -dimension in the second dimension of the array. For 16K systems, the length times the width must be less than 1400. The PMODE must be the same as that used for PUT.
10. The file-number can be 1 to 7. If it is 6, it specifies the screen, and the data are taken from the position where the cursor is. In this case, the values are those used in the COLOR statement.

**GOSUB***Format*

GOSUB line-number

The GOSUB statement transfers control to a subroutine beginning at the specified line-number. This need not be executable, nor the first statement of the subroutine. To return control to the statement following the GOSUB, a RETURN statement must be used.

*Example*

```

100 GOSUB 1000
200      (control returns here)
      *
      *
      *
1000      (subroutine)
      *
      *
      *
2000 RETURN

```

*See: ON GOSUB, POP, RETURN.*

## GOSUB

System		In	Format	Notes	Alternate Commands
APPLE	Integer	X		1, 3	
	Applesoft	X		2	
	DOS	X		2	
	Microsoft	X			
IBM	Cassette	X			
	Disk	X			
	Advanced	X			
TRS Mod III	Level I	X			
	Extended	X			
	Disk	X			
TRS Color	Level I	X			
	Extended	X			
	Disk	X			
Commodore	VIC 20	X			
ATARI	400/800	X		4	
ANSI	Minimum	X			

*Notes*

1. Maximum nesting is 16 deep.
2. Maximum nesting is 25 deep.
3. The line-number can be an arithmetic-expression.
4. Control returns to the *line* following the GOSUB, not the statement.

**GOTO***Format*

GOTO line-number

The GOTO statement transfers control to the specified line-number. This line need not be executable. If it is not, control passes to the first executable line after it. If the line contains more than one statement, control is passed to the first statement on the line. A space can optionally be put between GO and TO.

*Example*

```
100 INPUT A$
120 IF A$ = " " GOTO 100
```

The program fragment above waits for a nonnull string to be typed in.

*See: ON GOTO.*

GOTO
------

System		In	Format	Notes	Alternate Commands
APPLE	Integer	X		1	
	Applesoft	X			
	DOS	X			
	Microsoft	X			
IBM	Cassette	X			
	Disk	X			
	Advanced	X			
TRS Mod III	Level I	X			
	Extended	X			
	Disk	X			
TRS Color	Level I	X			
	Extended	X			
	Disk	X			
Commodore	VIC 20	X			
ATARI	400/800	X		1	
ANSI	Minimum	X			

*Note*

1. The line-number can be an arithmetic-expression.

## GR

*Format*

GR screen-number [,color-number]

The GR statement clears the screen and sets it to the low-resolution graphics mode. The screen number must be an integer with a value of 0 or 1. The default value is zero, which specifies a 40 × 40 screen with four lines of text; a value of 1 specifies a 40 × 48 screen with no text. The color number must be an integer between 0 and 15; the default is 0 (black).

*See: color codes, GRAPHICS, HGR, HGR2, SCREEN.*

GR

System		In	Format	Notes	Alternate Commands
APPLE	Integer	X		1	
	Applesoft	X		1	
	DOS	X		1	
	Microsoft	X			
IBM	Cassette				
	Disk				
	Advanced				
TRS Mod III	Level I				
	Extended				
	Disk				
TRS Color	Level I				
	Extended				
	Disk				
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

*Note*

1. GR cannot have any parameters. It sets the screen to the low-resolution graphics mode ( $40 \times 40$  with four lines of text). The screen is cleared, and COLOR is set to 0.

## GRAPHICS *Format*

### GRAPHICS arithmetic-expression

The GRAPHICS statement clears the screen and sets it to one of nine graphics modes. Modes 0, 1, and 2 are text modes; modes 3 to 8 are graphics modes. This statement also opens the screen as device number 6. The expression must be positive with a value between 0 and 8. A fractional value is rounded.

Modes 1 to 8 are split-screen modes. To override, add "+16" to the expression: GRAPHICS X + 16. If "+32" is added, it suppresses the clearing of the screen.

The expression must be positive with a value between 0 and 8. A fractional value is rounded.

The following table gives the characteristics of the modes:

Mode	Number of x Positions	Number of y Positions Split Screen	Number of y Positions Full Screen	Number of Colors	RAM Needed
0	40		24	2	993
1	20	20	24	5	513
2	20	10	12	5	261
3	40	20	24	4	273
4	80	40	48	2	537
5	80	40	48	4	1017
6	160	80	96	2	2025
7	160	80	96	4	3945
8	320	160	192	2	7900

GRAPHICS is supported only by ATARI BASIC.

*See: color codes, GR, HGR, HGR2, SCREEN.*

**HCOLOR***Format*

HCOLOR = arithmetic-expression

HCOLOR sets the high-resolution graphics color to the value specified. Once set, this color is not changed by HGR, HGR2, or RUN. HCOLOR does not affect the low-resolution graphics color.

The expression must have a value between 0 and 7. The values and their colors are:

0	Black-1	4	Black-2
1	Green	5	Depends on the TV
2	Blue	6	Depends on the TV
3	White-1	7	White-2

White-1 and -2 and black-1 and -2 are thicker lines.

*See: COLOR, HGR, HGR2, POSITION, SCRN, SETCOLOR.*

HCOLOR
--------

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft	X			
	DOS	X			
	Microsoft	X		1	
IBM	Cassette				
	Disk				
	Advanced				
TRS Mod III	Level I				
	Extended				
	Disk				
TRS Color	Level I				
	Extended				
	Disk				
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

*Note*

1. The expression can be between 0 and 12. For the colors that these values represent, see "color codes."

**HEX\$***Format*

HEX\$ (arithmetic-expression)

The hexadecimal function, HEX\$, returns as its value a string that represents the hexadecimal value of the expression. The expression must have a decimal value of between 0 and 65535, inclusive.

*Example*

```
100 Y$ = HEX$(X)
120 PRINT X, Y$
```

**Output for Various Values of X**

X	Y
1	1
100	64
132	84
32767	7FFF
65535	FFFF

*See: conversion tables, OCT\$.*

HEX\$
-------

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft	X		1	
IBM	Cassette	X		1	
	Disk	X		1	
	Advanced	X		1	
TRS Mod III	Level I				
	Extended				
	Disk				
TRS Color	Level I				
	Extended	X			
	Disk	X			
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

*Note*

1. The expression is rounded to an integer first.

### hexa- decimal conversion tables

*See: conversion tables.*

### hexa- decimal digit

A hexadecimal digit is one used to represent numbers in the hexadecimal (base 16) system. The digits, together with their decimal equivalents, are as follows:

Hex	Decimal
0 to 9	0 to 9
A	10
B	11
C	12
D	13
E	14
F	15

### hexa- decimal number

A number preceded by a “&H” is interpreted as a hexadecimal number. A hexadecimal number is one that uses the base 16 system. Consequently, each position in the number represents a multiple of a power of 16.

*Example*

The hex number 1234 represents

$$\begin{aligned}
 &1 \times 16^3 + 2 \times 16^2 + 3 \times 16^1 + 4 \times 16^0 \\
 &= 1 \times 4096 + 2 \times 256 + 3 \times 16 + 4 \times 1 \\
 &= 4660 \text{ in decimal}
 \end{aligned}$$

Values of hex numbers can range from 0000 to FFFF.

*See: binary number, conversion tables, octal number.*

hexadecimal numbers

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS	X		1	
	Microsoft	X			
IBM	Cassette	X			
	Disk	X			
	Advanced	X			
TRS Mod III	Level I				
	Extended				
	Disk	X			
TRS Color	Level I				
	Extended	X			
	Disk	X			
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

*Note*

1. The dollar sign (\$) is used before a number to indicate hexadecimal values.

**HGR***Format 1***HGR***Format 2*

**HGR** [screen-number] [,color]

The HGR statement is used to put the screen into high-resolution graphics mode.

*Format 1*

The HGR statement clears the screen to black and sets it to the high-resolution graphics mode (280 × 160), leaving four lines for text at the bottom of the screen. Then page 1 (8 to 16K) of high-resolution graphics memory is displayed.

This command does not affect HCOLOR or text screen memory. The text window is left at full screen but only the four bottom lines are displayed. The cursor remains in the text window, but it may not be visible.

After HGR has been executed, the screen can be set to full graphics mode by executing POKE – 16302,0 or POKE 49234,0. If HGR is subsequently executed, the screen resets to the high resolution and text mode.

### Format 2

This form of HGR sets the screen to one of four modes and, optionally, clears it. The following chart gives the actions as a function of the screen number, which must be between 0 and 3.

Screen Number	Clear Screen?	Mode
0	Yes	$280 \times 160 + 4$ lines text
1	Yes	$280 \times 192$
2	No	$280 \times 160 + 4$ lines text
3	No	$280 \times 192$

If no screen number is specified, 0 is assumed.

The color must be a number between 0 and 12. These colors are different from the usual ones associated with the values. See the entry on color codes for their meaning. If no color is specified, 0 (black) is used. If a color is specified with modes 0 or 1, the screen is filled with that color.

**See: GR, HCOLOR, HGR2, HIMEM:, LOMEM:.**

HGR
-----

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft	X	1		
	DOS	X	1		
	Microsoft	X	2		
IBM	Cassette				
	Disk				
	Advanced				
TRS Mod III	Level I				
	Extended				
	Disk				
TRS Color	Level I				
	Extended				
	Disk				
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

**HGR2***Format***HGR2**

The HGR2 statement clears the screen to black and puts it in full-screen high-resolution graphics mode (280 × 192). It then displays page 2 (16 to 24K) of graphics memory. HCOLOR and text screen memory are not affected.

After HGR2 has been executed, the screen can be set to mixed graphics and text mode by executing POKE -16301,0 or POKE 49233,0. If this is done, the four lines of text will be from page 2, not page 1, however.

HGR2 is supported only in Applesoft and Apple DOS.

*See: HCOLOR, HGR, HIMEM:, LOMEM:.*

**hierarchy  
of  
operations**

*See: arithmetic operations, logical functions.*

**HIMEM:***Format*

HIMEM: arithmetic-expression

HIMEM: is a variable that sets the address of the highest memory location available to the program, protecting the area above it. It is automatically set to the highest address available when BASIC is invoked.

The expression must be between -65535 and +65535; positive and negative values are considered equivalent. HIMEM: must be set higher than LOMEM:. If it is set above actual memory (e.g., to 65535 in a 16K machine) programs may not execute reliably. HIMEM: is not reset by CLEAR, RUN, NEW, and DEL. It is reset by (RESET + CONTROL-B), which also erases any stored program.

HIMEM: is supported only by Applesoft and Apple DOS.

*See: FREE, LOMEM:*

**HLIN***Format*

HLIN      starting-x-coordinate, ending-x-coordinate  
            AT y-coordinate

The **HLIN** statement plots a horizontal line from the starting to ending *x* points at the specified *y*-coordinate.

Both *x*-values must be between 0 and 39; the *y*-value must be between 0 and 47. The starting value must not exceed the ending value. If the screen is in text mode or in mixed graphics and text mode, then if the *y*-value is between 40 and 47, a line of characters is plotted where the dots would have been. This statement has no visible effect in high-resolution graphics mode.

*See: HPlot, LINE, Plot, VLin.*

HLIN
------

System		In	Format	Notes	Alternate Commands
APPLE	Integer	X			
	Applesoft	X		1	
	DOS	X		1	
	Microsoft	X		1	
IBM	Cassette				LINE
	Disk				LINE
	Advanced				LINE
TRS Mod III	Level I				
	Extended				
	Disk				
TRS Color	Level I				LINE
	Extended				LINE
	Disk				LINE
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

*Note*

1. In low-resolution graphics mode the color is determined by the most recently executed **COLOR** statement.

## HOME

*Format*

### HOME

The **home** statement moves the cursor to the upper left position within the scrolling window and clears all text in the window.

HOME

System		In	Format	Notes	Alternate Commands
APPLE	Integer				CALL -936
	Applesoft	X			
	DOS	X			
	Microsoft	X		1	
IBM	Cassette				CLS
	Disk				CLS
	Advanced				CLS
TRS Mod III	Level I				
	Extended				CLS
	Disk				CLS
TRS Color	Level I				CLS
	Extended				CLS
	Disk				CLS
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

*Note*

1. If a nonstandard terminal is connected, a "clear screen" sequence of characters is sent.

**HPLOT***Format 1*

HPLOT [x-coordinate-1, y-coordinate-1]  
 [TO x-coordinate-2, y-coordinate-2]...

*Format 2*

HPLOT TO x-coordinate, y-coordinate

*Format 1*

This form of the HPLOT statement draws a high-resolution line or point. If only the first set of  $x$  and  $y$  values is specified, a point is plotted at those coordinates. The color of the point is that specified by the last HCOLOR command.

If only the second set of  $x$  and  $y$  values is specified, a line is drawn from the last point plotted to the specified point. The color is determined by the last point plotted even if a HCOLOR statement has been executed since the last point. If no previous point has been plotted, no line is drawn.

If both sets of values are specified, a line is drawn from the first point to the second. If subsequent TO clauses are specified, lines are drawn from the last point plotted to the new location. The color is that specified by the most recent HCOLOR command.

HPLOT must be preceded by HGR or HGR2 to avoid wiping out the program and/or its variables.

All x-values must be between 0 and 279, inclusive; y-values, between 0 and 191, inclusive. If the mixed graphics and text mode is in effect, y-values from 160 to 191 will have no visible effect.

### Format 2

This form of HPLOT draws a line from the last point plotted to the point specified by the x and y coordinates. The color is determined by HCOLOR; if no HCOLOR statement has been executed, color 0 is used. This statement has no effect unless there has been a point plotted.

**See: DRAWTO, HCOLOR, HGR, HGR2, HLIN, PLOT, VLIN.**

### HPLOT

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft	X	1		
	DOS	X	1		
	Microsoft	X	1, 2	1	
IBM	Cassette				
	Disk				
	Advanced				
TRS Mod III	Level I				
	Extended				
	Disk				
TRS Color	Level I				
	Extended				
	Disk				
Commodore	VIC 20				
ATARI	400/800				DRAWTO
ANSI	Minimum				

### Note

1. If no color has been established, color 0 (black) is used.

**HSCRN***Format*HSCRN (*x*-value, *y*-value)

The HSCRN function tests a point on a high-resolution screen.

If a dot exists at the specified point, HSCRN returns a  $-1$ ; otherwise, it returns a value of zero. Unlike SCRN, HSCRN does not recognize the color of the point. The *x*-value must be between 0 and 279, inclusive; the *y*-value, between 0 and 191, inclusive.

HSCRN is supported only by Microsoft.

*See: SCRN.*

**HTAB***Format*

HTAB arithmetic-expression

The HTAB statement moves the cursor relative to the left edge of the window, but independently of the line width. That is, the cursor can be moved past the right-hand window edge. The cursor can be moved left or right.

The expression must be between 0 and 255. Values from 1 to 40 denote the current line; values from 41 to 80, the next line; and so on.

*See: LOCATE, PRINT, SPC, TAB, VTAB.*

HTAB
------

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft	X		3	
	DOS	X		3	
	Microsoft	X		1, 2	LOCATE
IBM	Cassette				LOCATE
	Disk				LOCATE
	Advanced				
TRS Mod III	Level I				
	Extended				
	Disk				
TRS Color	Level I				
	Extended				
	Disk				
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

*Notes*

1. The cursor is moved to an absolute location, not a relative one.
2. The cursor stays on the same line; the expression, which must be between 1 and 255, is taken modulo 40.
3. HTAB 0 moves the cursor to position 256.

## IF

### *Format*

```
IF expression|relation THEN line-number|statement-1
    [ELSE statement-2]
```

The IF statement evaluates a condition and, depending on the result of this evaluation, performs one or the other of two actions.

The operation is as follows. The relation or expression is evaluated. If it is true (that is, not equal to zero), then if a line-number is specified, control passes to that line number; if a statement-1 is specified, it is executed and then control passes to the next line in the program. If the relation is false (that is, equal to zero), then if the ELSE clause is not specified, control passes to the next line in the program; if the ELSE clause is specified, statement-2 is executed and then control passes to the next line in the program. If only a variable is specified, as in IF X THEN ..., it is the same as writing IF X > 0 THEN ...

The action of the IF statement can be summarized as follows:

	Condition True	Condition False
No ELSE clause	THEN clause executed and control passes to next line	Next line executed
ELSE clause specified	THEN clause executed and control passes to next line	ELSE clause executed and control passes to next line

In some implementations, either statement can be a group of statements. When they are to be executed, they are all executed as a group. (Obviously, if one of the statements causes a transfer of control, the remainder may not be executed.)

*Note:* When used with floating-point values, instead of a strict equality it is better to use inequalities. For example, the test `IF A = 0` will evaluate as false unless A is exactly zero. Often the result of a computation gives a small positive or negative value that is, for all practical purposes, zero. To accommodate this situation use `IF ABS(A) <= DELTA`, where DELTA is defined as appropriately small.

### *Example*

Assume that all variables start out with a value of zero. Executing

```
IF X > 10 THEN Y = 1 ELSE Z = 1
```

results in the following values of Y and Z for the given values of X:

X	Y	Z
5	0	1
10	0	1
15	1	0

Upon executing

```
100 IF X > 10 THEN Y = 1 : A = 1 ELSE Z = 1 : B = 1
120 C = 1
```

the following values of Y, A, Z, B, and C result for the given values of X:

X	Y	A	Z	B	C
5	0	0	1	1	1
10	0	0	1	1	1
15	1	1	0	0	1

*See: loop.*

IF

System		In	Format	Notes	Alternate Commands
APPLE	Integer	X		1,2,3,9	
	Applesoft	X		1,2,3,4,6	
	DOS	X		1,2,3,4,6	
	Microsoft	X		2,7	
IBM	Cassette	X		2,7	
	Disk	X		2,7	
	Advanced	X		2,7	
TRS Mod III	Level I	X		2,6	
	Extended	X		2,5	
	Disk	X		2,5	
TRS Color	Level I	X		6,7	
	Extended	X		6,7	
	Disk	X		6,7	
Commodore	VIC 20	X		6,8	
ATARI	400/800	X		2,6,7	
ANSI	Minimum	X		6	

### Notes

1. If the relation is false, control passes to the next statement, not necessarily the next line.
2. If a GO TO statement follows THEN, the THEN can be omitted.
3. If the variable before THEN ends in an "A", parsing problems occur. For example, IF A THEN ... parses to IF AT HEN ...
4. If a string is specified, it evaluates as nonzero (true) even if it is null or blank. Only the literal null string ("") evaluates as zero (false). But if the expression is a string variable and the previous statement assigned the null string to *any* string variable, the string evaluates to zero. If more than two statements of the form

IF string-variable THEN ...

are in the program, an error occurs.

5. A colon can appear before ELSE.
6. No ELSE clause is permitted.
7. The THEN clause can have several statements separated by colons. These statements will all be executed if the relation is true. However, a colon cannot immediately precede ELSE.
8. The line-number or statement after THEN must be on the same line.
9. The expression must be arithmetic; strings cannot be compared.

**IMP***Format*

argument-1 IMP argument-2

The implication function IMP is a logical function of two arguments. It has a value of false if its first argument is false and its second argument is true, and a value of true otherwise. In this function, the order of the arguments is important.

The arguments can be relations, logical variables, or anything that can be evaluated as true or false.

In formal logic, the implication function is “If P, then Q.”

Truth Table for IMP

p	q	p IMP q
F	F	T
F	T	T
T	F	F
T	T	T

If IMP is not implemented, it can be calculated by

$$\text{DEF FNIMP (P, Q) = (NOT P OR Q)}$$

*See: logical functions.*

IMP
-----

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft	X			
IBM	Cassette	X			
	Disk	X			
	Advanced	X			
TRS Mod III	Level I				
	Extended				
	Disk				
TRS Color	Level I				
	Extended				
	Disk				
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

**IN #***Format*

IN # slot-number

The IN # statement selects input from the peripheral attached to the specified slot. This determines which source is used to provide input for subsequent INPUT statements.

The slot-number must be between 1 and 7. If there is no peripheral in the designated slot, the system hangs. (Use RESET + CONTROL-C to recover.) Executing IN # 0 returns to normal keyboard input; slot 0 is not addressable by the program.

*See: INPUT #, PR #.*

IN #
------

System		In	Format	Notes	Alternate Commands
APPLE	Integer	X			
	Applesoft	X			
	DOS	X		1	
	Microsoft				INPUT #
IBM	Cassette				INPUT #
	Disk				INPUT #
	Advanced				INPUT #
TRS Mod III	Level I				
	Extended				INPUT #
	Disk				INPUT #
TRS Color	Level I				INPUT #
	Extended				INPUT #
	Disk				INPUT #
Commodore	VIC 20				INPUT #
ATARI	400/800				INPUT #
ANSI	Minimum				

*Note*

1. This statement must be executed, preceded by a CONTROL-D, as part of a PRINT statement:

```
PRINT CHR$(4);"IN # 6"
```

**increment**

An increment is a positive value that is added to a counter or variable. If the value is negative, or if a positive value is subtracted from the counter, it is called a decrement.

*See: decrement, FOR, loop.*

**INKEY\$***Format*

string-variable = INKEY\$

The INKEY\$ function has as its value the one-character string that corresponds to the last key struck on the keyboard. Characters input by this function are not normally displayed on the screen. If no key has been struck, a null string is returned. Once the key has been read in, the keyboard is reset, so the next INKEY\$ will wait for another key.

*Example*

To wait for any key to be struck:

```
100 A$ = INKEY$
120 IF A$ = " " THEN GOTO 100
140 (a key has been struck)
```

*See: GET.*

INKEY\$
---------

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				GET
	DOS				GET
	Microsoft	X		1	GET
IBM	Cassette	X		2	
	Disk	X		2	
	Advanced	X		2	
TRS Mod III	Level I				
	Extended	X			
	Disk	X			
TRS Color	Level I	X			
	Extended	X			
	Disk	X			
Commodore	VIC 20				GET
ATARI	400/800				
ANSI	Minimum				

*Notes*

1. CONTROL-C will not be passed through.
2. A 1- or 2-byte string can be returned. If the first byte is null (ASCII

0), the second byte contains a code which defines the key as follows:

3	NUL
15	SHIFT + TAB
16–25	ALT + Q, W, E, R, T, Y, U, I, O, or P
30–38	ALT + A, S, D, F, G, H, J, K, or L
44–50	ALT + Z, X, C, V, B, N, or M
59–68	Function keys F1 to F10 (if disabled as soft keys)
71	HOME
72	Cursor up
73	Pg Up
75	Cursor left
77	Cursor right
79	END
80	Cursor down
81	Pg Dn
82	INS
83	DEL
84–93	Uppercase + F1 through F10
94–103	CTL + F1 through F10
104–113	ALT + F1 through F10
114	CTL + Prtsc
115	CTL + Cursor left
116	CTL + Cursor right
117	CTL + END
118	CTL + Pg Dn
119	CTL + HOME
120–131	ALT + 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, —, =
132	CLT + Pg Up

### *Example*

To test and act on two-character codes:

```

100 X$ = INKEY$ : IF X$ = " " GOTO 100
120 IF LEN(X$) = 2 THEN      (two-code processing)
140 (one-code processing)
```

## **INP**

### *Format*

INP (port)

The INP statement returns a byte value from the specified port.

*See: OUT.*

INP

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft				
IBM	Cassette	X		1	
	Disk	X		1	
	Advanced	X		1	
TRS Mod III	Level I				
	Extended	X		2	
	Disk	X		2	
TRS Color	Level I				
	Extended				
	Disk				
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

*Notes*

1. The port can be between 0 and 65535; this is equivalent to the "IN =" assembly language instruction.
2. The port must be an integer between 0 and 255.

**INPUT***Format*

INPUT [;] ["message";] {variable,} ...

The INPUT statement accepts values from the keyboard and assigns them to the variables in the list. Both numeric and string variables can be input, and they can appear in any order in the variable list.

If a message is specified, it must be in quotes and followed by a semicolon. The message is printed, followed by a "?", and then the input is accepted. If no message is specified, only the "?" is printed.

The values being input must match the corresponding variables with respect to type (string versus numeric). If more values than there are variables are input, the excess values are ignored.

String data being input can optionally be enclosed in quotes. If string data begin with a quote, any character, even a comma or colon, can appear in the string. Leading and trailing spaces within the quotes are considered part of the string.

If string data do not begin with a quote, the comma and colon are not accepted as part of the string, since they will function as delimiters. Also, leading and trailing spaces are ignored. However, the string can contain a quote in any position except the first nonspace character.

*See: IN #, LINE INPUT, ON ERROR, RESUME.*

INPUT
-------

System		In	Format	Notes	Alternate Commands
APPLE	Integer	X		1, 5	
	Applesoft	X		2, 5, 6	
	DOS	X		2, 5, 6	
	Microsoft	X		3, 4	
IBM	Cassette	X		3, 5	
	Disk	X		3, 4	
	Advanced	X		3, 4	
TRS Mod III	Level I	X		5	
	Extended	X		5, 7	
	Disk	X		5, 7	
TRS Color	Level I	X		5	
	Extended	X		5	
	Disk	X		5	
Commodore	VIC 20	X		5	
ATARI	400/800	X		8	
ANSI	Minimum	X		5	

### Notes

1. A comma must be written after the message, not a semicolon.
2. If a message is present, no "?" is printed. A CONTROL-X and CONTROL-M cannot be in the string typed in. IF an ON ERR GOTO is used and the error-handling routine uses a GOTO to return to the INPUT statement, the eighty-sixth input error will cause a return to the monitor. This does not happen if RESUME is used in the error-handling routine.

Elements can be separated by a Carriage Return or a space. If the routine is awaiting a numeric input and a Return is pressed, it prompts again. If it is waiting for a string input, a Return causes a null string to be input. Excess values input are ignored.

3. The input is not accepted until all the values are good. If too many elements are input, it causes an error. If a comma follows the message instead of a semicolon, the "?" is not printed. Carriage Returns cannot be used to separate elements; only commas can be used. If a single item is being INPUT, a Carriage Return by itself causes a zero or null to be entered. A colon can be in a string that does not have quotes.
4. If INPUT is followed by a semicolon, a Return/Line Feed is not echoed in response to a Return typed by the user. The cursor remains on the same line as the response.
5. A semicolon cannot follow INPUT.
6. In numeric input, spaces are ignored in any position; so the string 1 2 3 4 is taken as 1234.
7. If no value is assigned to a variable, it keeps its old value.
8. A message cannot be specified; a PRINT statement must be used. String variables cannot be subscripted; array variables cannot be specified.

**INPUT #***Format*

INPUT # file-number, {variable} ...

The INPUT # statement takes input from the specified file and assigns it to the variables in the list.

The data elements should have the same representation as if they were being typed in. For numeric values, leading spaces, Line Feeds, and Carriage Returns are ignored; the first character that is not one of the three is taken as the start of the number. A space, comma, Carriage Return, or Line Feed terminates the number.

For string data, if the first character is a quote, leading and trailing spaces, commas, and colons within the quotes are accepted as part of the string. If the first character is not a quote, leading and trailing spaces are ignored and the string ends when a comma, colon, Carriage Return, or Line Feed is encountered, or after 255 characters or an end of file (EOF) is read.

The file-number can be from 1 to 15.

*See: IN #, INPUT, PR #, PRINT #.*

INPUT #
---------

System		In	Format	Notes	Alternate Commands
APPLE	Integer				IN #
	Applesoft				IN #
	DOS				IN #
	Microsoft	X		1	
IBM	Cassette	X		2	
	Disk	X		2	
	Advanced	X		2	
TRS Mod III	Level I				
	Extended	X		3	
	Disk	X			
TRS Color	Level I	X		3	
	Extended	X		3	
	Disk	X			
Commodore	VIC 20	X		4	
ATARI	400/800	X		5	
ANSI	Minimum				

*Notes*

1. Works only with sequential disk files.
2. Works on a sequential disk or cassette file.
3. A buffer-number is used instead of a file-number. It must be  $-1$ , which is the cassette. The variable list must be configured identically to that in the PRINT# statement that wrote the tape (with respect to the number and types of variables and their order in the list). The actual names need not be the same.
4. Input is read up to a Carriage Return.
5. The file number can be 1 to 7.

**INPUT\$***Format*

INPUT\$ (arithmetic-expression [, [#] file-number])

The INPUT\$ statement obtains a string of characters from the specified file. The expression specifies the number of characters returned. If the file is associated with the keyboard, no characters are displayed on the screen and all control characters, except CONTROL-BREAK, are passed through.

The file-number must be between 0 and 15, inclusive; 0 specifies the keyboard.

*See: IN#, INPUT, INPUT#.*

INPUT\$
---------

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft	X		1	
IBM	Cassette	X			
	Disk	X			
	Advanced	X			
TRS Mod III	Level I				
	Extended				
	Disk				
TRS Color	Level I				
	Extended				
	Disk				
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

*Note*

1. A CONTROL-C will not pass through if input from the keyboard.

## INSTR

*Format*

INSTR ([position,] string-1, string-2)

The INSTR statement searches string-1 for occurrences of string-2. The search begins at the character in string-1 specified by the position.

The first occurrence of string-2 that is fully contained in string-1 is located and its starting position in string-1 is returned. If the string is not found, 0 is returned.

The position must be an integer between 1 and 255; if it is not specified, 1 is used. If the starting position is greater than the number of characters in string-1, or if string-1 is null, a 0 is returned.

Example

	Value of X:
100 A\$ = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"	
120 B\$ = "CDE"	
140 C\$ = "CDG"	
160 N\$ = " "	
180 X = INSTR(A\$,B\$)	3
200 X = INSTR(3,A\$,B\$)	3
220 X = INSTR(4,A\$,B\$)	0
240 X = INSTR(A\$,N\$)	1
260 X = INSTR(5,A\$,N\$)	5
280 X = INSTR(A\$,C\$)	0
300 X = INSTR(30,A\$,N\$)	0

See: LEFT\$, MID\$, RIGHT\$.

INSTR					
System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft	X			
IBM	Cassette	X			
	Disk	X			
	Advanced	X			
TRS Mod III	Level I				
	Extended				
	Disk	X			
TRS Color	Level I				
	Extended	X			
	Disk	X			
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

INT

Format

INT (arithmetic-expression)

The integer function, INT, returns the largest integer that is not greater than the expression. INT differs from FIX in that for a negative number the

value returned can be less than the argument. The expression is not limited to the usual range for integers.

*Example*

$$100 Y = \text{INT}(X)$$

### Output for Different Values of X, Compared with FIX

X	INT(X)	FIX(X)
-2.0	-2	-2
-2.1	-3	-2
-2.9	-3	-2
+2.0	+2	+2
+2.1	+2	+2
+2.9	+2	+2

*See: CINT, FIX.*

INT
-----

System		In	Format	Notes	Alternate Commands
APPLE	Integer			1	
	Applesoft	X		2	
	DOS	X		2	
	Microsoft	X			
IBM	Cassette	X			
	Disk	X			
	Advanced	X			
TRS Mod III	Level I	X			
	Extended	X			
	Disk	X			
TRS Color	Level I	X			
	Extended	X			
	Disk	X			
Commodore	VIC 20	X			
ATARI	400/800	X			
ANSI	Minimum	X			

*Notes*

1. This is a command that erases the current program.
2. This is also a command that switches to integer BASIC.

**integer**

In BASIC, an integer is a whole number that is not stored as a floating-point value. Integers occupy 2 bytes and have values between  $-32768$  and  $+32767$ . An integer is defined by a “%” following the name or by having the first letter of its name appear in a DEFINT statement. Note that a single- or double-precision variable can have an integer *value*, but they are not considered integers. An integer constant is one written without a decimal point.

If floating-point variables are not needed, using integers can speed up program execution.

*See: CINT, DEFINT, FIX, INT, real number.*

integer
---------

System		In	Format	Notes	Alternate Commands
APPLE	Integer	X		1, 2	
	Applesoft	X		2	
	DOS	X			
	Microsoft	X			
IBM	Cassette	X			
	Disk	X			
	Advanced	X			
TRS Mod III	Level I	X			
	Extended	X			
	Disk	X			
TRS Color	Level I			3	
	Extended			3	
	Disk			3	
Commodore	VIC 20	X			
ATARI	400/800	X		3	
ANSI	Minimum				

*Notes*

1. As this BASIC supports only integers, the percent sign cannot be used.
2. The smallest negative number is  $-32767$ .
3. There are no integers per se, only floating-point variables with integer values.

**INVERSE***Format***INVERSE**

The INVERSE statement sets the video mode such that subsequent output shows as black on white. It does not affect characters currently on the

screen when the command is executed, nor characters typed into the computer.

*See: FLASH, NORMAL.*

INVERSE
---------

System	In	Format	Notes	Alternate Commands
APPLE	Integer			POKE 50,127
	Applesoft	X		
	DOS	X		
	Microsoft	X		
IBM	Cassette			
	Disk			
	Advanced			
TRS Mod III	Level I			
	Extended			
	Disk			
TRS Color	Level I			SHIFT + Ø
	Extended			
	Disk			
Commodore	VIC 20		1	
ATARI	400/800			
ANSI	Minimum			

*Note*

1. From the keyboard, use CONTROL-RVS + ON; from the program use POKE 36879, PEEK (36879) AND 247.

## inverse function

If we consider a function as a transformation or mapping of one set of values onto another set, the inverse of a function is a function that performs the reverse transformation.

For example, when one writes  $Y = \text{LOG}(X)$ , the LOG function transforms the value of  $X$  into a new value, which becomes the value of  $Y$ . The inverse of the LOG function, EXP, will transform this value of  $Y$  back into the original  $X$ . In other words,  $\text{EXP}(\text{LOG}(X)) = X$  for any valid value of  $X$ .

Note, however, that due to round-off error the final value of  $X$  may differ slightly from the original value. For example, if  $X = 9$ ,  $\text{EXP}(\text{LOG}(X))$  may evaluate to 8.999999.

The ASC and CHR\$ functions are also inverses. For any value,  $X$ ,  $\text{ASC}(\text{CHR}\$(X))$  equals  $X$  itself.

*See: function.*

# J

## JOYSTK

*Format*

JOYSTK (arithmetic-expression)

The joystick function, JOYSTK, returns a value from 0 to 63 that indicates one of the coordinates of a joystick.

The expression must have a value between 0 and 3; these values signify:

- 0 Left joystick, horizontal position
- 1 Left joystick, vertical position
- 2 Right joystick, horizontal position
- 3 Right joystick, vertical position

The upper leftmost position is 0,0; the upper rightmost is 0,63; the lower leftmost is 63,0; and the lower rightmost is 63,63.

JOYSTK (0) must be executed before any of the other coordinates are read or erroneous results may be obtained.

*See: BUTTON, PADDLE, PDL, PTRIG, STICK, STRIG.*

### JOYSTK

System		In	Format	Notes	Alternate Commands
APPLE	Integer				PDL
	Applesoft				PDL
	DOS				PDL
	Microsoft				PDL
IBM	Cassette				STICK
	Disk				
	Advanced				
TRS Mod III	Level I				
	Extended				
	Disk				
TRS Color	Level I	X			
	Extended	X			
	Disk	X			
Commodore	VIC 20				
ATARI	400/800				STICK, PADDLE
ANSI	Minimum				

**K** K is an abbreviation for “kilo,” a prefix indicating 1000. However, when used to refer to the capacity of a storage device (memory, cassette, or disk) it means a multiple of 1024 bytes; thus “32K” means 32,768 bytes.

*See: byte.*

## KEY

*Format 1*

KEY integer, string-variable

*Format 2*

KEY ON|OFF|LIST

*Format 3*

KEY (integer) ON|OFF|STOP

*Format 1*

This form of the KEY statement assigns a text string to a function key. The integer, which must be between 1 and 10, inclusive, specifies the function key. The string-variable contains the text to be assigned to the key. Once the KEY command has been executed, pressing the key causes the text to be input to the system as if it had been typed in. Assigning text to a key erases any previous assignment.

*Format 2*

This form of KEY controls the display of the text currently assigned to the function keys. If ON is specified, the first six characters assigned to each of the keys is displayed on the twenty-fifth line of the screen. If OFF is specified, this display is erased, but the keys are not disabled. If LIST is specified, all the text associated with each key is displayed.

*Format 3*

This form of KEY enables or disables trapping of the function and cursor movement keys. The integer must be between 1 and 14, where values from 1

to 10 refer to the function keys and values of 11 to 14 to the cursor movement keys:

11	Cursor up
12	Cursor left
13	Cursor right
14	Cursor down

ON activates trapping of the specified key. When trapping is activated, if an ON KEY GOSUB statement is executed, whenever the designated key is pressed a transfer is made to the subroutine. OFF disables trapping; if a key is pressed, the system does not remember the event. STOP disables trapping, but if a key is pressed the event is remembered and, when an ON KEY GOSUB is subsequently executed, the trap occurs immediately.

All formats of KEY are supported only by IBM BASIC (all levels).

*See: ON KEY*

**keyword**      *See: reserved word.*

**KILL**          *Format*

KILL file-specification

The KILL command deletes the specified file from the disk. The extension must be included in the file specification and the file must be closed.

*Example*

```
100 A$ = "MYFILE.BAS"
120 KILL A$
      or
120 KILL "MYFILE.BAS"
```

*See: DELETE.*

KILL
------

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				DELETE
	Microsoft	X			
IBM	Cassette				
	Disk	X			
	Advanced	X			
TRS Mod III	Level I				
	Extended				
	Disk	X		1	
TRS Color	Level I				
	Extended				
	Disk	X		1	
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

*Note*

1. The file-specification must be in quotes.

# L

## **last point referenced**

Graphics instructions usually operate with respect to a point that is defined in the statement. In some instructions, if no point is specified, the starting point is the last point plotted on the screen. This is called the “last point referenced.” Another term is “last point plotted.”

In IBM applications, all graphics locations specified in relative form are taken with respect to the last point referenced.

*See: CIRCLE, DRAW, LINE, PAINT, PSET, PRESET, relative form.*

## **least significant digit**

The least significant digit is the one that contributes the smallest value to the magnitude of a number. (This is not necessarily indicative of the precision of the number.)

When dealing with binary numbers, the least significant *bit* (LSB) is the rightmost bit in the number.

*See: most significant digit.*

## **LEFT\$**

*Format*

LEFT\$(string-variable, arithmetic-expression)

The LEFT\$ function has as its value the leftmost characters of the string, as determined by the expression. The value of the expression must be between 0 and 255. If the expression is greater than the number of characters in the string, the function's value is the entire string. If the value is zero, the null string is returned.

*Example*

```
100 A$ = "ABCDEFGH"  
120 B$ = "ABC"  
140 X$ = LEFT$(A$,2)  
160 Y$ = LEFT$(B$,5)  
180 Z$ = LEFT$(B$,0)  
200 PRINT X$, Y$, Z$
```

**Output**

AB      ABC      (Z\$ is null)

*See: INSTR, MID\$, null string, RIGHT\$.*

LEFT\$
--------

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft	X		1	
	DOS	X		1	
	Microsoft	X			
IBM	Cassette	X			
	Disk	X			
	Advanced	X			
TRS Mod III	Level I				
	Extended	X			
	Disk	X			
TRS Color	Level I	X			
	Extended	X			
	Disk	X			
Commodore	VIC 20	X			
ATARI	400/800				
ANSI	Minimum				

*Note*

1. The value of the expression can be between 1 and 255.

**LEN***Format*

LEN (string-expression)

The LEN function has as its value the number of characters in the expression, including nonprinting characters. The length of the null string is 0.

*Example*

```

100 A$ = "ABC"
120 B$ = "DEF"
140 C$ = " "
160 X = LEN(A$)
180 Y = LEN(A$ + B$)

```

```
200 Z = LEN (C$)
220 PRINT X, Y, Z
```

### Output

```
3    6    0
```

LEN
-----

System		In	Format	Notes	Alternate Commands
APPLE	Integer	X		1	
	Applesoft	X		2	
	DOS	X		2	
	Microsoft	X			
IBM	Cassette	X			
	Disk	X			
	Advanced	X			
TRS Mod III	Level I				
	Extended	X			
	Disk	X			
TRS Color	Level I	X			
	Extended	X			
	Disk	X			
Commodore	VIC 20	X			
ATARI	400/800	X			
ANSI	Minimum				

### Notes

1. The argument cannot be an expression, only a string variable.
2. The function returns 0 to 255. If the number of characters is over 255 (as in a concatenated expression), it causes an error.

## LET

### Format

[LET] variable = expression

The LET statement assigns the value of the expression to the variable. The keyword LET is not required in personal computer BASIC, but should be included if compatibility with older versions is required. This is also called the assignment statement.

*See: SWAP*

LET

System		In	Format	Notes	Alternate Commands
APPLE	Integer	X			
	Applesoft	X			
	DOS	X			
	Microsoft	X			
IBM	Cassette	X			
	Disk	X			
	Advanced	X			
TRS Mod III	Level I	X			
	Extended	X			
	Disk	X			
TRS Color	Level I				
	Extended	X			
	Disk	X			
Commodore	VIC 20	X			
ATARI	400/800	X			
ANSI	Minimum	X		1	

*Note*

1. LET is required in the statement.

## LINE

*Format*

LINE [(x-coordinate-1, y-coordinate-1)] —  
 (x-coordinate-2, y-coordinate-2),  
 PSET|PRESET|[color] [,B|BF]

The LINE statement draws a line or a rectangle. The line is drawn from the first set of x and y coordinates to the second. If the first set of values is not specified, the last point plotted is used as the starting point. If no previous point has been plotted, point (128,96) is used. After the statement has executed, the last point referenced is the one specified by the second set of coordinates.

PSET specifies that the foreground color is used for the line; PRESET, the background color.

B draws a box instead of a line. The first set of values specify the upper left-hand corner, the second set of points, the lower right-hand corner. BF not only draws a box, but it also fills it with the foreground color.

*See: DRAW, HLIN, HPLOT, PLOT, VLIN.*

LINE
------

System		In	Format	Notes	Alternate Commands
APPLE	Integer				HLIN, VLIN
	Applesoft				HLIN, VLIN
	DOS				
	Microsoft				
IBM	Cassette	X		1	
	Disk	X		1	
	Advanced	X		1	
TRS Mod III	Level I				
	Extended				
	Disk				
TRS Color	Level I				
	Extended	X		2	
	Disk	X		2	
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

*Notes*

1. This can be used in graphics mode only. The coordinates can be in relative or absolute form. If the second set of values is relative, it is considered relative to the first set, not to the last point plotted. A coordinate that is negative is taken as zero; a coordinate greater than the maximum is taken as the maximum value: 199 for  $y$ , 639 for  $x$  (high resolution). In medium resolution, an  $x$ -value over 319 wraps to the next line. PSET and PRESET cannot be specified.
2. A color cannot be specified, only PSET and PRESET.

**line***See: program line.***LINE  
INPUT***Format 1*

LINE INPUT #file-number, string-variable

*Format 2*

LINE INPUT [;] ["character-string";] string-variable

The LINE INPUT statement inputs an entire line into a string variable.

*Format 1*

This form of the LINE INPUT statement reads a line from the specified file into the variable. A line is defined as all characters until a Carriage Return, an end of file (EOF), or the 255th character is encountered. If a Line Feed/Carriage Return is met, it is passed; however, a Carriage Return/Line Feed terminates the input. The file-number must be between 1 and 15, inclusive.

*Format 2*

This form of LINE INPUT reads a line from the keyboard into the specified variable. All characters, including leading spaces, are read in until a CR (ASCII 13) is encountered. If a character-string is specified, it is printed before the line is accepted; if no string is specified, nothing, not even the system prompt, is output prior to accepting input.

**See: INPUT, IN#, READ.**

LINE INPUT

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft	X	1, 2	1	
IBM	Cassette	X	1, 2	4, 5	
	Disk	X	1, 2	3, 5	
	Advanced	X	1, 2	3, 5	
TRS Mod III	Level I				
	Extended				
	Disk	X	1, 2	2, 4	
TRS Color	Level I				
	Extended	X	2	2, 4	
	Disk	X	1	2, 4	
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

*Notes*

1. A CONTROL-C returns to the command level. If CONT is subsequently issued, the instruction starts all over again.
2. Uses a buffer-number instead of a file-number.

3. In Format 2, if the semicolon is included after INPUT, the user's Carriage Return will not generate a Carriage Return/Line Feed and the cursor will remain on the same line as the input.
4. A semicolon cannot be used after INPUT.
5. A CONTROL-BREAK returns to the command level. If CONT is subsequently issued, the instruction starts all over again.

## line number

The following chart gives the valid values of line numbers for the various systems.

*See: program line.*

				line number	
System		In	Format	Notes	Range (Zero to)
APPLE	Integer				32767
	Applesoft				63999
	DOS				63999
	Microsoft			1	65529
IBM	Cassette			1	65529
	Disk			1	65529
	Advanced			1	65529
TRS Mod III	Level I				65529
	Extended			1	65529
	Disk			1	65529
TRS Color	Level I				63999
	Extended				63999
	Disk				63999
Commodore	VIC 20				63999
ATARI	400/800				32767
ANSI	Minimum				9999

### Note

1. The period (.) can be used as a symbol for the current line in LIST, DELETE, and so on.

## LIST

### Format

LIST [line-number-1] [—] [line-number-2]

The LIST command displays one or more lines of the program currently in memory.

If no line-numbers are specified, the entire program is listed. If only one line-number is specified, that line is listed. If two line-numbers are specified, all lines from the first to the second, inclusive, are listed.

If only line-number-1 and the hyphen are specified, all lines from that number to the end of the program are listed. If only the hyphen and line-number-2 are specified, all lines from the first line of the program to that number are listed.

*See: ENTER, LLIST.*

LIST

System		In	Format	Notes	Alternate Commands
APPLE	Integer	X		1, 2	
	Applesoft	X		1, 3	
	DOS	X		1, 3	
	Microsoft	X		3, 4	
IBM	Cassette	X		4, 5	
	Disk	X		4, 5	
	Advanced	X		4, 5	
TRS Mod III	Level I	X		1, 4	
	Extended	X		4	
	Disk	X		4	
TRS Color	Level I	X			
	Extended	X			
	Disk	X			
Commodore	VIC 20	X		1	
ATARI	400/800	X		2, 6	
ANSI	Minimum				

### Notes

1. Cannot appear in a program.
2. Must use a comma instead of a hyphen between line numbers.
3. Can use a comma or a hyphen between line numbers.
4. The period ( . ) can be used to represent the current line.
5. If the line numbers are followed with a file-specification, the output is to the specified file, not the screen.
6. If a file specification in quotes precedes the line numbers, the lines are written to the cassette. If the specification is just "P", output goes to the printer.

**literal**

A literal is a string of characters whose value is determined by the characters themselves. For example, consider the two statements

```
100 X$ = "AB$"  
200 X$ = AB$
```

In line 100, the value assigned to X\$ consists of the three characters A, B, and \$; whereas in line 200 the value assigned to X\$ is the *contents* of the string variable AB\$.

Similarly for numeric variables:

```
100 X = 12.34  
200 X = A1
```

In line 100, the value assigned to X is 12.34; but in line 200, X gets whatever value the variable A1 has.

In the first example, the string "AB\$" is a literal—its value is that of the characters that make up the string—whereas the value of the *variable* AB\$ depends on its current contents.

In the second example, the value assigned to X is that represented by the characters themselves—the numeric value 12.34. Note that this is not the same as writing X\$ = "12.34"; in this case the value of X\$ is the five characters in the literal "12.34", and it has no numeric value.

There are both numeric and nonnumeric literals in BASIC. A numeric literal is also called a constant; a nonnumeric literal is sometimes called simply a string.

In input/output statements, depending on the rules, string literals can be written with or without quotation characters around them. For an example, see INPUT.

*See: string variable.*

**LLIST**

*Format*

```
LLIST [line-number-1] [-] [line-number-2]
```

The LLIST command operates identically to the LIST command except that the output goes to the printer instead of the screen.

*See: LIST.*

LLIST
-------

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft	X		2	
IBM	Cassette	X			
	Disk	X		1	
	Advanced	X		1	
TRS Mod III	Level I				
	Extended	X			
	Disk	X			
TRS Color	Level I	X			
	Extended	X			
	Disk	X			
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

*Notes*

1. The command can be terminated by CONTROL-BREAK.
2. A printer must be in slot 1; it is assumed to be 132 characters wide. After the command has been executed, control returns to the command level.

**LOAD***Format 1*

LOAD file-specification [,R]

*Format 2*

LOAD [file-name] [,device]

*Format 3*

LOAD file-name [,Ddrive] [,Sslot] [,Vvolume]

The LOAD command loads a program from cassette or disk into memory.

*Format 1*

This form of LOAD loads the specified file. The file must be a BASIC program; if no extension is specified, "BAS" is assumed. If R is specified, the program is run after it is loaded.

*Format 2*

In this form of LOAD the device can be 1 for the cassette, or 8 for the disk. If no device is specified, the cassette is assumed. If no program-name is specified, the first program encountered is loaded.

Normally, the program will load at location 4096; but if the machine has the extra 3K of memory, it loads at location 1024. If the command parameter is 1, the program is loaded from the same location at which it was saved. The new program will overwrite the one currently in memory. If it is shorter, some variables will be preserved.

The asterisk (\*) can be used as a "wild card." The command

LOAD "\*",8

loads the first program on disk. The command

LOAD "AB\*",8

loads the first program whose name begins with AB.

*Format 3*

This form of LOAD is used only for disk. It erases memory, loads the specified file from disk, and then changes to the correct BASIC (integer or floating) for the program.

This command must be executed, preceded by a CONTROL-D, as part of a PRINT statement:

PRINT CHR\$(4); "LOAD MYFILE, D1, S6"

The allowable values for parameters and their defaults are shown below. Once a value has been changed, the last value specified is the default.

Parameter	Range	Default
drive	1 or 2	Drive 1
slot	1 to 7	Slot from which DOS was booted
volume	1 to 254	Volume from which DOS was booted

*See: CLOAD, CLOADM, ENTER, LOADM, SAVE.*

LOAD

System		In	Format	Notes	Alternate Commands
APPLE	Integer	X	1	1	
	Applesoft	X	1	1	
	DOS	X	3		
	Microsoft	X	1	4	
IBM	Cassette	X	1	2, 3, 4	
	Disk	X	1	3, 4	
	Advanced	X	1	3, 4	
TRS Mod III	Level I				
	Extended				
	Disk	X	1	4, 5	
TRS Color	Level I				
	Extended				
	Disk	X	1	5	
Commodore	VIC 20	X	2		
ATARI	400/800	X	1	5, 6	
ANSI	Minimum				

*Notes*

1. A program-name cannot be specified, only LOAD. No checks are made to see if the cassette is connected or ready, so it must be running before the command is executed. A tone is sounded before and after loading.
2. If no device is specified, CAS1:, the only valid device, is used.
3. If CAS1: is the device, then if a program name is not specified, the next program on tape is loaded.
4. If R is specified, all open files are left open; if it is not specified, they are closed.
5. The file-specification must be in quotes.
6. The R cannot be specified, and no assumption is made about the extension.

**LOADM***Format*

LOADM "file-name" [,offset]

The LOADM command loads a machine language program from the disk and then executes it. The program is loaded at the address that was specified when it was saved. If an offset is specified, it is considered a hexadecimal number and is added to the program addresses, thereby relocating the program. (However, any absolute address in the program will not be changed.)

The file-name must be in quotes; if no extension is specified, BIN is assumed.

LOADM is supported only by TRS Color DOS.

*See: CLOAD, CLOADM, LOAD, SAVEM.*

## LOC

### Format

LOC (file-number)

The LOC function has as its value the number of the last record accessed in the specified buffer. LOC is valid only if the file is open and a GET has been executed for it.

The file-number must be between 1 and 15, inclusive.

*See: GET, LOF, OPEN.*

LOC
-----

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft	X		2	
IBM	Cassette				
	Disk	X		1	
	Advanced	X		1	
TRS Mod III	Level I				
	Extended				
	Disk	X		3	
TRS Color	Level I				
	Extended				
	Disk	X		3	
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

### Notes

1. If the file is sequential, LOC returns the number of records read from or written to the file since it was opened. For communications files, LOC returns the number of bytes in the input buffer waiting to be read. If there are over 255 bytes, LOC returns 255. If the file is open for sequential input and no input has been performed, LOC returns a 1.

2. With sequential files, the value returned is the number of sectors read or written since the file was opened. (One sector is 128 bytes.)
3. A buffer-number must be used instead of a file-number.

**LOCATE***Format 1*

LOCATE [row] [,column] [,cursor] [,start] [,stop]

*Format 2*

LOCATE x-coordinate, y-coordinate, arithmetic-variable

The LOCATE statement positions the cursor on the screen.

*Format 1*

In addition to positioning the cursor, this form of LOCATE also defines its size. The cursor is positioned to the specified row and column; if the cursor parameter is 0, the cursor is invisible, if 1, the cursor is visible.

The start and stop parameters define the size of the cursor. This size is specified in terms of *scan lines*, where the top scan line is 0, and the bottom scan line is 7 for the color/graphics monitor, and 13 for the monochrome display.

If *start* is specified and *stop* is not, *stop* is set equal to *start*. If *start* is greater than *stop*, wraparound occurs.

Valid values for the parameters are:

row	1 to 25
column	1 to 40 or 80
cursor	0 or 1
start	0 to 13
stop	0 to 13

If any parameter is not specified, the current value is used.

*Format 2*

This form of LOCATE positions the cursor to the specified location and reads the characteristics of that location, storing it in the specified variable.

The range of values that this can return depends on the mode of the screen.

Mode	Range
0, 1, 2	0 to 255
3, 5, 7	0 to 3
4, 6, 8	0 or 1

*See: CSRLIN, POSITION, SCRIN.*

LOCATE
--------

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft				
IBM	Cassette	X	1		
	Disk	X	1		
	Advanced	X	1		
TRS Mod III	Level I				
	Extended				
	Disk				
TRS Color	Level I				
	Extended				
	Disk				
Commodore	VIC 20				
ATARI	400/800	X	2		
ANSI	Minimum				

**LOCK**

*Format*

LOCK file-name [,Ddrive] [,Sslot] [,Vvolume]

The LOCK command protects a file from being deleted, renamed, or altered.

This command must be executed, preceded by a CONTROL-D, as part of a PRINT statement:

PRINT CHR\$(4); "LOCK MYFILE, D1, S6"

The allowable values for parameters and their defaults are shown below. Once a value has been changed, the last value specified is the default.

Parameter	Range	Default
drive	1 or 2	Drive 1
slot	1 to 7	Slot from which DOS was booted
volume	1 to 254	Volume from which DOS was booted

LOCK is supported only by Apple DOS.

*See: file specification, UNLOCK.*

## LOF

*Format*

LOF (file-number)

The LOF function has as its value the highest numbered record of the specified file. The file can be sequential or random access; it must be open and, if it is in extend mode, a GET must be executed before LOF is valid.

The file-number must be between 1 and 15, inclusive.

*See: GET, LOC, OPEN.*

LOF

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft	X		1	
IBM	Cassette				
	Disk	X		1, 2	
	Advanced	X		1, 2	
TRS Mod III	Level I				
	Extended				
	Disk	X		3	
TRS Color	Level I				
	Extended				
	Disk	X		3	
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

*Notes*

1. The value returned is the number of bytes allocated to the file. For disk files, the value returned is a multiple of 128.
2. For communications files, the number of bytes in the input buffer is returned.
3. A buffer-number must be used instead of a file-number.

**LOG***Format*

LOG (arithmetic-expression)

The natural logarithm function, LOG, returns as its value the natural logarithm of the expression. It is the inverse of the EXP function. The expression must be greater than zero.

If  $X = \text{LOG}(Y)$ , then  $e^X = Y$ .

*Rules of Logarithms*

$$\text{LOG}(X * Y) = \text{LOG}(X) + \text{LOG}(Y)$$

$$\text{LOG}(X/Y) = \text{LOG}(X) - \text{LOG}(Y)$$

$$\text{LOG}(X^Y) = Y * \text{LOG}(X)$$

To convert a natural logarithm to a base 10 logarithm, divide it by LOG(10). One can define the following function:

*Example*

```
100 DEF FNLN(X) = LOG(X)/LOG(10)
120 K = 100
140 Y = LOG(K)
160 Z = FNLN(K)
180 PRINT LOG(2.718283), Y, Z
```

**Output**

```
1      4.60517      2
```

*See: CLOG, EXP.*

LOG

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft	X			
	DOS	X			
	Microsoft	X			
IBM	Cassette	X			
	Disk	X			
	Advanced	X			
TRS Mod III	Level I				
	Extended	X			
	Disk	X			
TRS Color	Level I				
	Extended	X			
	Disk	X			
Commodore	VIC 20				
ATARI	400/800	X			
ANSI	Minimum	X			

## logical functions

Logical functions deal with relations and with variables that can have one of two values: *true* or *false*. (These are called logical variables.) False is represented by zero, true by any nonzero value. In general, when the system software returns a true value it is  $-1$ .

The two logical functions implemented most often are AND and OR; each is a function of two arguments. The logical function NOT, a function of one argument, is also usually implemented.

De Morgan's theorem defines a special relationship between the AND and OR functions:

$$\text{NOT } (p \text{ AND } q) = \text{NOT}(p) \text{ OR } \text{NOT}(q)$$

$$\text{NOT } (p \text{ OR } q) = \text{NOT}(p) \text{ AND } \text{NOT}(q)$$

Three additional functions are implemented in some systems: the equivalence function, EQV; the implication function, IMP; and the exclusive or function, XOR.

Logical functions are evaluated in this order:

NOT, AND, OR, XOR, IMP, EQV

If an expression contains two instances of the same function, evaluation is from left to right. Parentheses can be used to override the order listed above, the innermost set of parentheses being evaluated first, then the next innermost, and so on.

*See: AND, EQV, IMP, NOT, OR, XOR.*

logical functions

System		Notes	NOT	AND	OR	EQV	IMP	XOR
APPLE	Integer		X	X	X			
	Applesoft	1	X	X	X			
	DOS		X	X	X			
	Microsoft		X	X	X	X	X	X
IBM	Cassette		X	X	X	X	X	X
	Disk		X	X	X	X	X	X
	Advanced		X	X	X	X	X	X
TRS Mod III	Level I		X	X	X			
	Extended		X	X	X			
	Disk		X	X	X			
TRS Color	Level I		X	X	X			
	Extended		X	X	X			
	Disk		X	X	X			
Commodore	VIC 20		X	X	X			
ATARI	400/800	1	X	X	X			
ANSI	Minimum							

*Note*

1. The system returns a 1 for true, not  $-1$ .

**logical variable**

A logical variable is a variable that can have a value of true or false. False is represented by zero, true by any nonzero value. In general, when the system software returns a true value, it is  $-1$ .

*See: logical functions.*

**LOMEM:** *Format*

LOMEM: arithmetic-expression

LOMEM: is a variable that is used to set the address of the lowest memory location available to the program. It is used to protect variables from high-resolution graphics pages. Before a program is executed, LOMEM: is automatically set to the end of the current program.

The expression must be between -65535 and 65535 (positive and negative values are considered equivalent). LOMEM: must be set lower than HIMEM:. Once set, LOMEM: cannot be set to a new value lower than its current value unless it is first reset. It can be set higher, however.

LOMEM: is reset by NEW, DEL, RESET + CONTROL-B, and by adding or changing a program line. It is *not* reset by RUN, RESET + CONTROL-C, or RESET ØG. If changed by the program it may cause parts of the program to be unavailable to itself, causing chaos.

If LOMEM: is set so as to point into the operating system or a stored program, an error occurs. The current location of LOMEM: is in locations 106 (high byte) and 105 (low byte). If set too low, an out-of-memory error occurs.

LOMEM: is supported only by Applesoft and Apple DOS.

*See: FREE, HIMEM:.*

## loop

A loop is one or more instructions that are executed for a specified number of times, or until a condition becomes true.

### *Example*

To clear two 10-element arrays, one could use the following loop:

```
100 FOR I = 1 TO 10
120 A$(I) = " "
140 SX(I) = 0
160 NEXT I
```

*See: FOR, WHILE.*

## LPOS

### *Format*

LPOS (arithmetic-expression)

The LPOS function has as its value the current logical position of the print head within the printer buffer.

In cassette BASIC, the argument is a dummy variable; for noncassette BASIC, the expression indicates:

0 or 1	Line printer 1
2	Line printer 2
3	Line printer 3

The position returned is not necessarily the physical position of the print head. If the width has been specified as infinite, the head position goes from 0 to 255 and then back to 0 again.

*Example*

```
100 Y = LPOS(1)
140 LPRINT "ABCDEFGHI"
140 Z = LPOS(1)
160 PRINT Y,Z
```

**Output**

```
1      10
```

*See: POS, WIDTH.*

LPOS
------

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft	X		1	
IBM	Cassette	X			
	Disk	X			
	Advanced	X			
TRS Mod III	Level I				
	Extended				
	Disk				
TRS Color	Level I				
	Extended				
	Disk				
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

*Notes*

1. The expression is always a dummy argument.

**LPRINT**

*Format*

LPRINT [expression [;[,]]...

LPRINT has the same format and operates the same as PRINT, but sends the output to the line printer. It can be used with any form of PRINT except PRINT@. If a TAB command is used with LPRINT, it can move the carriage only to the right.

*See: PRINT.*

LPRINT

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft	X		1	
IBM	Cassette	X		2	
	Disk	X		2	
	Advanced	X		2	
TRS Mod III	Level I				
	Extended	X			
	Disk	X			
TRS Color	Level I				
	Extended				
	Disk				
Commodore	VIC 20				
ATARI	400/800	X			
ANSI	Minimum				

### Notes

1. Assumes a 132-character-wide printer, which must be in slot 1.
2. Assumes an 80-character-wide printer.

## LSET

### Format

LSET field-name = character-string

The LSET command moves the character-string into the specified field, left-justified. If the character-string is smaller than the field, space fill occurs on the right; if it is larger, the excess is truncated.

The character-string can be a string-variable or a literal. Numeric values must be converted to strings before they can be LSET. (Note that a field-name is not the same as a normal string-variable.)

*See: assignment statement, FIELD, MKD\$, MKI\$, MKN\$, MKS\$, RSET.*

LSET

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft	X		1	
IBM	Cassette				
	Disk	X		1	
	Advanced	X		1	
TRS Mod III	Level I				
	Extended	X			
	Disk	X			
TRS Color	Level I				
	Extended				
	Disk	X			
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

*Note*

1. LSET can be used with a nonfielded variable to left-justify it. For example:

```

100 A$ = SPACE$(5)
120 N$ = "ABC"
140 LSET A$ = N$
160 PRINT ">"A$;"<"

```

Output: >ABC <

**machine language**

Machine language is a combination of 1's and 0's that are interpreted by the computer as instructions, data, and/or addresses.

**mask**

A mask is a pattern of bits that is used to test selected bits in a byte. The mask contains a one bit for each bit that is to be considered in the test and a zero bit for those that are not of interest. When the mask is ANDed with the byte being tested, all bits in the result that do not correspond to one bits in the mask are zero. Bits corresponding to one bits in the mask are left in their original state.

Often, one wishes to test for a certain pattern of bits. The byte to be tested is first XORed with the desired pattern and the result is then ANDed with the mask. If the result is zero the desired pattern is present; if the result is nonzero, the pattern is not present.

For example, assume that we are testing for a pattern of 1010 in the rightmost 4 bits of a byte, and we do not care what the leftmost 4 bits are.

Byte	1 1 0 0 1 0 1 0	
XOR	XXXX1010	(X = anything)
Result	XXXX0000	
AND	00001111	
Result	00000000	

If any bit does not match the pattern:

Byte	1 1 0 0 1 0 1 1
XOR	XXXX1010
Result	XXXX0001
AND	00001111
Result	00000001

**mathematical operations**

*See: arithmetic operations.*

**matrix functions**

BASIC was originally designed to aid college students in solving complicated mathematical problems. As such it contained a set of statements

designed to facilitate matrix manipulation. Most personal computers do not support matrix operations directly; to implement these function FOR ... NEXT or WHILE ... WEND loops must be used.

## **MAXFILES**    *Format*

### MAXFILES integer

MAXFILES specifies the number of files that can be active at one time. When executed, 595 bytes are reserved for each file.

The integer must be between 1 and 16, inclusive; the default is 3. Each DOS command, except PR #, IN #, and MAXFILES itself, requires a file buffer.

Executing MAXFILES in the immediate mode will erase an integer BASIC program that is in memory. This command cannot be executed from within an integer BASIC program unless you use an EXEC file.

For Applesoft programs, it should be the first statement in the program since it affects string variable space and changes the value of HIMEM:.

This command must be executed, preceded by a CONTROL-D, as part of a PRINT statement:

```
PRINT CHR$(4); "MAXFILES 5"
```

MAXFILES is supported only by Apple DOS.

*See: EXEC.*

## **MEM**    *Format*

### MEM

The MEM function has as its value the number of unused and unprotected bytes in memory.

*See: FRE, HIMEM:, LOMEM:.*

MEM

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				FRE
	DOS				FRE
	Microsoft				FRE
IBM	Cassette				FRE
	Disk				FRE
	Advanced				FRE
TRS Mod III	Level I	X			
	Extended	X			
	Disk	X			
TRS Color	Level I	X			
	Extended	X			
	Disk	X			
Commodore	VIC 20				FRE
ATARI	400/800				
ANSI	Minimum				

### memory pad mode

Memory pad mode is a mode in ATARI BASIC in which one can use the screen without disturbing the resident program.

### MERGE

*Format*

MERGE file-specification [,R]

The MERGE command loads a program file from disk and merges it with the program currently in memory. If R is specified, the resultant program is then run. In order to be merged, a program must have been saved with the ASCII option.

If a line number of the resident program is the same as a line number of the program being merged, the resident line is overwritten.

The file-specification can be a literal or in a string variable.

*See: CHAIN, SAVE.*

MERGE
-------

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft	X		1	
IBM	Cassette	X		1	
	Disk	X		1	
	Advanced	X		1	
TRS Mod III	Level I				
	Extended				
	Disk	X			
TRS Color	Level I				
	Extended				
	Disk	X		2	
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

*Notes*

1. The "R" is not permitted; if no extension is specified, BAS is assumed.
2. The file-specification must be in quotes.

**MID\$***Format*

MID\$ (string-variable, starting-position, length)

MID\$ is both a function and a statement. As a function its value is the characters of the specified string as defined by the starting-position and length. If the length is omitted, all of the string to the right of the starting-position is returned. If the length is zero or the starting-position is greater than the length of the string, a null string is returned. If the starting-position plus the length is greater than 255 or the length of the string, all of the string to the right of the starting position is returned.

*Example*

```

100 A$ = "ABCDEFGHJKLMNOP"
120 B$ = "DEF"
140 W$ = MID$(A$,7,4)
160 X$ = MID$(B$,2,4)
180 Z$ = MID$(A$,20,2)
200 PRINT W$, X$, Z$

```

**Output**

      GHIJ      EF      (Z\$ is null)

As a statement, MID\$ allows a string variable to have part of its contents changed.

*Example*

```
100 C$ = "ABCDEFGG"
120 MID$(C$,3,3) = "123"
140 PRINT C$
```

**Output**

AB123FG

*See: concatenation, INSTR, LEFT\$, RIGHT\$.*

MID\$
-------

System		In	Format	Notes	Alternate Commands
APPLE	Integer			1	
	Applesoft	X		2	
	DOS	X		2	
	Microsoft	X			
IBM	Cassette	X			
	Disk	X			
	Advanced	X			
TRS Mod III	Level I				
	Extended	X		2	
	Disk	X		2	
TRS Color	Level I	X		2	
	Extended	X		2	
	Disk	X		2	
Commodore	VIC 20	X		2	
ATARI	400/800				
ANSI	Minimum				

*Notes*

1. The notation X\$(5,7) indicates the fifth through seventh characters of X\$. One can write Y\$ = X\$(5,7) to assign these characters to Y\$. One cannot write X\$(5,7) = "ABC", however.
2. Function only.

**mini-floppy disks** *See: disks.*

**MKD\$** *Format*

MKD\$ (arithmetic-expression)

The MKD\$ function treats the expression as a double-precision number and converts it to an 8-byte character string for subsequent FIELDing. This is the inverse function of CVD.

*See: CVD, FIELD, GET, LSET, PUT, RSET.*

MKD\$

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft	X			
IBM	Cassette				
	Disk	X			
	Advanced	X			
TRS Mod III	Level I				
	Extended				
	Disk	X			
TRS Color	Level I				
	Extended				
	Disk				
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

**MKI\$** *Format*

MKI\$ (arithmetic-expression)

The MKI\$ function treats the expression as an integer and converts it to a 2-byte character string for subsequent FIELDing. This is the inverse function of CVI.

The expression must have a value between - 32768 and + 32767.

*See: CVI, FIELD, GET, LSET, PUT, RSET.*

MKI\$

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft	X		1	
IBM	Cassette				
	Disk	X		1	
	Advanced	X		1	
TRS Mod III	Level I				
	Extended				
	Disk	X		2	
TRS Color	Level I				
	Extended				
	Disk				
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

*Notes*

1. The expression is rounded to an integer.
2. The expression is truncated to an integer.

**MKN\$***Format*

MKN\$ (arithmetic-expression)

The MKN\$ function converts the expression to a 5-byte coded string for subsequent storage in a formatted disk file. This is the inverse function of CVN.

MKN\$ is supported only by TRS Color DOS.

*See: CVN, FIELD, GET, LSET, PUT, RSET.*

**MKS\$***Format*

MKS\$ (arithmetic-expression)

THE MKS\$ function treats the expression as a single-precision number and converts it to a 4-byte character string for subsequent FIELDing. This is the inverse function of CVS.

*See: CVS, FIELD, GET, LSET, PUT, RSET.*

MKS\$

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft	X			
IBM	Cassette				
	Disk	X			
	Advanced	X			
TRS Mod III	Level I				
	Extended				
	Disk	X			
TRS Color	Level I				
	Extended				
	Disk				
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

**MOD***Format*

arithmetic-expression-1 MOD arithmetic-expression-2

The modulus operator, MOD, divides the first expression by the second and has as its value the remainder.

*Example*

```
100 X = 10 MOD 6
120 Y = 6 MOD 10
140 Z = 12345 MOD 123
160 PRINT X, Y, Z
```

**Output**

```
4      6      45
```

This can be implemented by

$$\text{FNMD}(A,B) = (\text{INT}(A) - \text{INT}(\text{INT}(A)/\text{INT}(B)) * \text{INT}(B))$$

*See: arithmetic operations.*

MOD

System		In	Format	Notes	Alternate Commands
APPLE	Integer	X			
	Applesoft				
	DOS				
	Microsoft	X			
IBM	Cassette	X			
	Disk	X			
	Advanced	X			
TRS Mod III	Level I				
	Extended				
	Disk				
TRS Color	Level I				
	Extended				
	Disk				
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

**MON***Format*

MON [C] [,I] [,O]

The MON command enables monitoring of certain operations. The parameters signify:

- C     Display all disk commands
- I     Display all disk input to the computer
- O     Display all computer output to the disk

The three parameters can be written in any order; at least one must be specified. The monitoring remains in effect until a NOMON, boot, restart, or change of language (Integer to Floating, or vice versa) occurs.

MON is supported only by Apple DOS.

*See: NOMON.*

**most  
significant  
digit**

The most significant digit is the one which contributes the largest value to the number in which it appears. This is not necessarily related to the precision of the number.

When dealing with binary numbers, the most significant bit (MSB) is the leftmost bit, regardless of its value.

*See: least significant digit.*

**MOTOR***Format 1*

MOTOR ON|OFF

*Format 2*

MOTOR [arithmetic-expression]

The MOTOR statement turns the cassette motor on and off.

*Format 2*

If the expression is zero, the motor is turned off; if it is nonzero, the motor is turned on. If no state is specified, the motor is set to the state opposite its present state.

MOTOR

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft				
IBM	Cassette	X	2		
	Disk	X	2		
	Advanced	X	2		
TRS Mod III	Level I				
	Extended				
	Disk				
TRS Color	Level I	X	1		
	Extended	X	1		
	Disk	X	1		
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

**multiple statements**

Most BASIC Systems allow more than one statement on a line, subject to the maximum line length. Often, this reduces the storage requirements of the program.

Multiple statements on a line are separated by colons. Depending on the implementation, statements such as IF may operate differently when multiple statements are used after the THEN or ELSE clauses.

## NAME

*Format 1*

NAME file-specification AS file-name

*Format 2*

NAME [line-number] [,starting-line] [,increment]

*Format 1*

This form of the NAME command changes the name of a disk file. The file whose name is in the file-specification has its name changed to the file-name in the AS clause. (A file with this name cannot currently exist on the disk.) If no device is specified in the file-specification, the current drive is used.

*Format 2*

This form of NAME renumbers the lines in a program by first changing the value of starting-line to the specified line-number; subsequent lines in the program are then assigned the value of the preceding line plus the increment.

If the starting-line is not specified, the entire program is renumbered. If either the line-number or increment is not specified, 10 is used.

*See: RENUM.*

NAME
------

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft	X	1		
IBM	Cassette				
	Disk	X	1		
	Advanced	X	1		
TRS Mod III	Level I				
	Extended	X	2		
	Disk	X	2		
TRS Color	Level I				
	Extended				
	Disk				
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

**name**

A variable name must begin with a letter. The number of subsequent characters allowed varies with the implementation, as shown below.

**IBM and MicroSoft**

Names can be any length; however, only the first 40 characters are significant. Reserved words can be embedded in names.

**Apple**

Names must have fewer than 100 characters. Reserved words cannot be embedded in names.

**Applesoft**

A name can have up to 238 characters, but only the first two characters are significant. Reserved words cannot be embedded in names.

**TRS Mod III**

Names can be any length; however, only the first two characters are significant. Reserved words cannot be embedded in names.

**TRS Color**

Names can have only two characters, both of which must be alphabetic.

**Commodore**

Names can have up to 255 characters; only the first two are significant. The first character must be alphabetic, the others can be alphanumeric. Reserved words cannot be embedded in the name.

**ATARI**

Names can have up to 120 characters. Only capital letters and digits can be used. Only 128 names can be used per program.

**ANSI**

Names can have only two characters; two letters or a letter and a digit.

*See: reserved word.*

**NEW**

*Format*

NEW

The NEW command clears memory of both variables and program, and closes all open disk files. Control then returns to the command level.

*See: CLEAR.*

NEW
-----

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft	X			
	DOS	X			
	Microsoft	X			
IBM	Cassette	X			
	Disk	X			
	Advanced	X			
TRS Mod III	Level I	X			
	Extended	X			
	Disk	X			
TRS Color	Level I	X			
	Extended	X			
	Disk	X			
Commodore	VIC 20	X			
ATARI	400/800	X		1	
ANSI	Minimum				

*Note*

1. Can be used only in direct mode.

## NEXT

*Format*

NEXT [counter]...

The NEXT statement delimits a FOR loop. The counter is optional; if it is not specified, the default is the counter associated with the nearest active FOR. If there is a list of counters, the innermost loop's counter must be written first.

*See: FOR, loop.*

NEXT
------

System		In	Format	Notes	Alternate Commands
APPLE	Integer	X		1	
	Applesoft	X			
	DOS	X			
	Microsoft	X			
IBM	Cassette	X			
	Disk	X			
	Advanced	X			
TRS Mod III	Level I	X			
	Extended	X			
	Disk	X			
TRS Color	Level I	X		1	
	Extended	X		1	
	Disk	X		1	
Commodore	VIC 20	X		1	
ATARI	400/800	X		1	
ANSI	Minimum	X		1	

*Note*

1. A counter must be specified.

## nibble

A nibble is half a byte (4 bits), when this entity is directly addressable.

*See: bit, byte.*

## NOMON

*Format*

NOMON [C] [,I] [,O]

The NOMON command disables the specified monitoring functions.

- C    Disable monitoring of disk commands
- I    Disable monitoring of disk input to the computer
- O    Disable monitoring of computer output to the disk

The three parameters can be written in any order; at least one must be specified.

NOMON is supported only by Apple DOS.

*See: MON.*

**NORMAL** *Format***NORMAL**

The **NORMAL** statement sets the screen mode such that subsequent input and output shows as white on black. It does not affect characters currently on the screen when the command is executed.

*See: FLASH, INVERSE.*

NORMAL
--------

System		In	Format	Notes	Alternate Commands
APPLE	Integer				POKE 50,255
	Applesoft	X			
	DOS	X			
	Microsoft	X			
IBM	Cassette				
	Disk				
	Advanced				
TRS Mod III	Level I				
	Extended				
	Disk				
TRS Color	Level I				SHIFT + Ø
	Extended				
	Disk				
Commodore	VIC 20			1	
ATARI	400/800				
ANSI	Minimum				

*Note*

1. From the keyboard use CONTROL-RVS + OFF. From a program use POKE 36879, PEEK (36879) OR 8

**NOT** *Format***NOT** argument

**NOT** is a logical function of one argument. It has as its value the negation of its argument, as shown in the following table.

Truth Table for NOT

p	NOT p
F	T
T	F

The argument can be a logical variable, a relation, or anything that can be evaluated as true or false.

*Example*

```
IF NOT A = B THEN ...  
IF NOT (A = B) THEN ...
```

The first example compares the value of NOT A with B. The second example is equivalent to IF A > < B THEN.

NOT is supported by all BASIC systems.

*See: AND, logical functions, OR.*

**NOTE**

*Format*

NOTE #disk, arithmetic-variable-1, arithmetic-variable-2

The NOTE statement stores the current sector and byte locations of the specified disk in the first and second variables, respectively. The next disk operation will be done starting at this location. The disk parameter can be an arithmetic expression.

NOTE is supported only by ATARI BASIC.

*See: DSKI\$, DSKO\$, POINT.*

**NOTRACE**

*Format*

NOTRACE

The NOTRACE command disables the trace function.

*See: TRACE, TROFF, TRON.*

NOTRACE

System		In	Format	Notes	Alternate Commands
APPLE	Integer	X			
	Applesoft	X			
	DOS	X			
	Microsoft	X			
IBM	Cassette				TROFF
	Disk				TROFF
	Advanced				TROFF
TRS Mod III	Level I				
	Extended				TROFF
	Disk				TROFF
TRS Color	Level I				TROFF
	Extended				TROFF
	Disk				TROFF
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

**null string**

A null string is a string that contains no characters. Since a null string is completely empty, it is not the same as a string of one or more blanks or spaces.

As a literal, a null string consists of two adjacent quotation characters. The length of a null string, as returned by LEN, is zero. Because in set theory the null set is unique, sometimes one speaks of “the null string.”

**numeric character**

A numeric character is any one of the digits 0 through 9.

*See: alphabetic character, alphanumeric character.*

# O

## OCT\$

*Format*

OCT\$ (arithmetic-expression)

The OCT\$ has as its value a string that represents the octal value of the expression. The expression is rounded to an integer before the operation.

*Example*

```
100 Y$ = OCT$(78)
120 PRINT Y$, OCT$(8.3)
```

**Output**

```
116      123
```

(Note that Y\$ is a string, not a numeric value.)

*See: HEX\$*

OCT\$
-------

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft	X			
IBM	Cassette	X			
	Disk	X			
	Advanced	X			
TRS Mod III	Level I				
	Extended				
	Disk				
TRS Color	Level I				
	Extended				
	Disk				
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

## octal conversion tables

*See: conversion tables.*

## octal number

A number preceded by a "&O" or "&" is interpreted as an octal value. An octal number can range in value from 0 to 177777 (base 8).

*Example*

&0123 or &123 is interpreted as decimal 83.

*See: binary number, hexadecimal number.*

octal numbers

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft	X			
IBM	Cassette	X			
	Disk	X			
	Advanced	X			
TRS Mod III	Level I				
	Extended				
	Disk	X			
TRS Color	Level I				
	Extended	X			
	Disk	X			
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

## ON COM

*Format*

ON COM (buffer-number) GOSUB line-number

The ON COM statement enables trapping of activity associated with a communications buffer. When data enter the specified buffer, control is transferred to the subroutine at the specified line-number. If the line-number is zero, this trapping is disabled; to reactivate, a COM ON statement must be executed.

When a trap occurs, an implicit COM STOP is executed, so recursive traps are not possible. Upon return from the trap routine an implicit COM ON is executed unless an explicit COM OFF was executed in the trap routine.

The buffer-number must be 1 or 2.

ON COM is supported only by IBM Advanced BASIC.

*See: COM, trap.*

## ON ERROR

*Format*

ON ERROR GO TO line-number

The ON ERROR command enables trapping of errors. When an error occurs, an unconditional transfer is made to the specified line-number; no error message is printed, and the machine does not halt. To be effective, this statement must be executed before the error occurs. To disable a previously set ON ERROR, an ON ERROR GO TO 0 must be executed.

*See: ERL, ERR, ERROR, RESUME, TRAP.*

### ON ERROR

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft	X		1, 2	
	DOS	X		1, 2	
	Microsoft	X		3	
IBM	Cassette	X		3	
	Disk	X		3	
	Advanced	X		3	
TRS Mod III	Level I				
	Extended	X			
	Disk	X			
TRS Color	Level I				
	Extended				
	Disk				
Commodore	VIC 20				
ATARI	400/800				TRAP
ANSI	Minimum				

*Notes*

1. ON ERR must be used instead of ON ERROR. The error code is in location 222. To disable the error-detection flag, use POKE 216,0.
2. In case of a FOR...NEXT or GOSUB...RETURN, the error-handling routine must restart the loop at the FOR or GOSUB, not at the NEXT or RETURN. If used to handle errors associated with a GET; two consecutive GET errors cause the program to hang (use RE-

SET + CONTROL = C to recover). If a GOTO ends the error-handling routine, there is no problem.

If used in trace mode or in a program that has a PRINT statement, after 43 errors occur (not necessarily consecutive) control returns to the monitor. If these errors are caused by an INPUT statement, and RESUME is used to leave the error-handling routine, all works well. If a GOTO ends the routine, the eighty-seventh INPUT error causes control to return to the monitor.

3. If the error-processing routine contains an ON ERROR GOTO 0 statement, the message for the error that caused the branch to the routine is printed. If an error occurs during an error-processing routine itself, the appropriate message is printed and control returns to the command level.

### **ON GOSUB** *Format*

ON arithmetic-expression GOSUB {line-number} ...

The expression is evaluated and converted to an integer. Control is then transferred to that subroutine whose line-number is in the ordinal position in the statement corresponding to the value of the expression.

The expression must be between 0 and 255. If it has a value of 0 or if its value exceeds the number of elements in the list, control passes to the next statement. If the expression is less than zero, an error occurs. The statement that exits the subroutine must be a RETURN.

ON GOSUB is supported by all BASICs except Apple Integer, which uses GOSUB with an arithmetic expression.

*See: GOSUB, RETURN.*

### **ON GOTO** *Format*

ON arithmetic-expression GOTO {line-number} ...

The expression is evaluated and converted to an integer. Control is then transferred to the line whose line-number is in the ordinal position in the statement corresponding to the value of the expression.

The expression must be between 0 and 255. If it has a value of zero or its value exceeds the number of elements in the list, control passes to the next statement and no transfer of control is performed. If the expression is less than zero, an error occurs.

ON GOTO is supported by all BASICs except Apple Integer, which uses GOTO with an arithmetic expression.

*See: GOTO.*

**ON KEY** *Format*

ON KEY (key-number) GOSUB line-number

The ON KEY statement is used to specify a line to which BASIC traps when a key is pressed. If the line-number is zero, this trapping is disabled.

The key-number must be an integer between 1 and 14, with the following meaning:

Value	Associated Key
1 to 10	Function keys 1 to 10
11	Cursor up
12	Cursor left
13	Cursor right
14	Cursor down

ON KEY is supported only by IBM Advanced BASIC.

*See: KEY.*

**ON PEN** *Format*

ON PEN GOSUB line-number

The ON PEN statement enables trapping of light-pen activity. When the light pen senses a point, control is transferred to the subroutine at the specified line-number. If the line-number is zero, this trapping is disabled.

ON PEN is supported only by IBM Advanced BASIC.

*See: PEN.*

**ON STRIG** *Format*

ON STRIG (stick-number) GOSUB line-number

The ON STRIG statement enables trapping of trigger activity. When a trigger on a joystick is pressed, control is transferred to the subroutine at the specified line-number. If the line-number is zero, this trapping is disabled.

The stick-number must be an integer with a value of 0, 2, 4, or 6, where 0 is button A1, 2 is button B1, 4 is button A2, and 6 is button B2.

ON STRIG is supported only by IBM Advanced BASIC.

*See: **BUTTON**, **STRIG**.*

### **one's complement**

The one's complement of a binary number is that number obtained by converting all the 0's in the number to 1's and all the 1's to 0's. Thus it is the same as the negation (NOT) of the binary string that represents the value.

*Example*

The one's complement of 00101110 is 11010001.

*See: **NOT**, **two's complement**.*

### **OPEN**

*Format 1*

OPEN "mode", #file-number, file-specification [record-length]

*Format 2*

OPEN file-number [,device][,subcommand][,file-name]

*Format 3*

OPEN file-name [,record-length][,Ddrive][,Sslot][,Vvolume]

*Format 4*

OPEN file-specification [FOR model] AS [#] file-name  
[LEN = record-length]

*Format 5*

OPEN #device-number, operation-code, auxiliary-code,  
file-specification

The OPEN statement prepares a file for input/output processing and associates a buffer with it.

*Format 1*

In this form of OPEN, the mode parameter must be in quotes; it can be any of the following:

Parameter	Meaning
I	Input mode, sequential file
O	Output mode, sequential file
E	Extend mode, sequential file
D	Direct file, input, or output mode
R	Random file, input, or output mode

The file-number must be between 1 and 15, inclusive.

*Format 2*

In this form of OPEN, the file-number must be an integer from 1 to 255; this is the number by which other BASIC statements reference the file.

The device must be one of the following:

0	Keyboard	1	Cassette
2	RS-232 Device	3	Screen
4, 5	Printer	8	Disk

The subcommand must be one of the following.

**Cassette**

0	Read
1	Write
2	Write with an end-of-tape (EOT) marker

**Disk**

1 to 14	Open data channel
15	Open command channel

**Keyboard/screen**

1 to 255	No effect
----------	-----------

**Printer**

0	Uppercase/graphics
7	Uppercase/lowercase

The statement

OPEN 1, 0	Reads the keyboard
OPEN 1, 1, 0, "name"	Reads from cassette
OPEN 1, 8, 15 "command"	Sends the command to the disk controller

### *Format 3*

This form of OPEN allocates a 595-byte buffer to the file and prepares the system for performing input/output operations on the file at its beginning. The file must be a text file. If the specified file does not exist, one is created. If any file with the same name is already open, that file is closed before the new one is opened.

This command must be executed, preceded by a CONTROL-D, as part of a PRINT statement:

```
PRINT CHR$(4); "OPEN MYFILE, D1, S6"
```

The allowable values for parameters and their defaults are shown below. Once a value has been changed, the last value specified is the default.

Parameter	Range	Default
record-length	1 to 32767	1
drive	1 or 2	Drive 1
slot	1 to 7	Slot from which DOS was booted
volume	1 to 254	Volume from which DOS was booted

For sequential files, the record length cannot be specified; for random access files, it must be specified. Each time a file is opened, the length must be the same as when the file was created.

### *Format 4*

The form of OPEN functions similarly to format 1, except that the mode must be one of the following:

OUTPUT	Output mode, sequential file.
INPUT	Input mode, sequential file.
APPEND	Same as EXTEND; the sequential file is positioned at its end; subsequent WRITE add data to the file. This can be used only with disk files.

If the mode is omitted, random access is assumed.

*Format 5*

The device number can be between 1 and 7. The operation code can have the following values:

- 4     Input
- 6     Disk directory input (in this case the file specification is the search specification)
- 8     Output
- 9     Append
- 12    Input/output

The file specification must be in quotes; any of the other parameters can be arithmetic expressions.

*See: APPEND, CLOSE, CMD, GET, INPUT#, PRINT#, PUT.*

OPEN
------

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS	X	3		
	Microsoft	X	1	1, 5, 6	
IBM	Cassette	X	1, 4	1, 6	
	Disk	X	1, 4	1, 6	
	Advanced	X	1, 4	1, 6	
TRS Mod III	Level I				
	Extended				
	Disk	X	1	2, 7	
TRS Color	Level I	X	1	3, 4, 7	
	Extended	X	1	3, 4, 7	
	Disk	X	1	3, 4, 7	
Commodore	VIC 20	X	2		
ATARI	400/800	X	5		
ANSI	Minimum				

*Notes*

1. Modes can be I, O, and R only. The record-length can be between 1 and 32767, inclusive; the default is 128.
2. The number sign (#) cannot be used. Mode D is not allowed. A record-length can be used only if the records are variable length; it can be between 1 and 256 inclusive. The default is 256.
3. For disk files, only modes I, O, and D are allowed. If no extension is specified, DAT is assumed. If the file is direct access, the record-

length can be specified; the default is 256. The buffer-number can be between 1 and 15, inclusive.

4. For nondisk files, only modes I and O are allowed, and the only buffer-numbers are -1 for the cassette and -2 for the printer. A record-length cannot be specified.
5. Only disk files are supported.
6. A file can be opened for sequential input or in random mode under more than one number. A sequential output file can be opened under only one number.
7. A buffer-number must be used instead of a file-number.

## OPEN COM

### *Format*

```
OPEN "COM buffer-number: [speed] [,parity] [,data] [,stop]
      [,RS] [,CS[value]] [,DS[value]] [,CD[value]] [,LF]"
      AS [#] file-number [LEN = length]
```

The OPEN COM statement opens a communication file in RS-232-compatible mode. The communication file is associated with the specified file-number. A communications device can be open to only one file at a time; the quotes are required where shown in the format.

The speed, parity, data, and stop parameters must appear in the order shown; the RS, CS, DS, CD, and LF parameters can appear in any order. The meaning of the various parameters is as follows:

The buffer-number must be an integer with a value of 1 or 2.

Speed must be an integer that specifies the transfer rate in baud. The valid values are 75, 110, 150, 300, 600, 1200, 1800, 2400, 4800, and 9600. The default is 300 baud.

Parity is a one-character literal; its values and their meaning are:

Character	Meaning
S	Parity bit is treated as a space (zero bit)
O	Odd parity
M	Parity bit is treated as a mark (one bit)
E	Even parity
N	No parity

Data is an integer constant that specifies the number of bits. Valid values are 4, 5, 6, 7, and 8. The default is 7 bits. If this parameter is 8, the parity must be "N"; if it is 4, the parity cannot be "N". For numeric data, this parameter must be 8.

Stop is an integer constant that specifies the number of stop bits. Its value must be 1 or 2. For 75 and 110 baud, the default is 2; for other speeds, it is 1. If the data parameter is 4 or 5, specifying a 2 for stop will result in  $1\frac{1}{2}$  bits actually being used.

**RS**

Normally, the Request to Send (RTS) line is activated when an OPEN COM is executed. Specifying RS suppresses this. If RS is specified, and CS is not, then CS0 becomes the default.

**CS**

Normally, if the Clear to Send (CTS) line is off, I/O operations to the file will fail. Specifying CS with no value or with a zero value causes the CTS line to be ignored. If a value is included, it specifies the number of milliseconds to wait for the signal before returning a device timeout error. (Normally, the wait is about 1 second.)

**DS**

Normally, if the Data Set Ready (DSR) line is off, I/O operations to the file will fail. Specifying DS with no value or with a zero value causes the DSR line to be ignored. If a value is included, it specifies the number of milliseconds to wait for the signal before returning a device timeout error. (Normally, the wait is about 1 second.)

**CD**

Normally, the Carrier Detect (CD or RLSD) line is ignored. If CD is specified, the line is tested. If a value is included, it specifies the number of milliseconds to wait before returning a device timeout error. (If the value is omitted or equal to zero, it has the same effect as not specifying CD.)

(In all of the parameters noted above, the value can be between 0 and 65535.)

**LF**

If LF is specified, a Line Feed (ASCII 10) is sent after a Carriage Return (ASCII 13). The INPUT# and LINE INPUT statements stop when they read a Carriage Return from a communications file opened with the LF option. The Line Feed is always ignored.

**file-number**

The file-number must be an integer between 1 and 15.

**LEN**

The length parameter specifies the maximum number of bytes that can be accessed in the communication buffer by a GET or PUT statement. The default is 128 bytes.

OPEN "COM is supported only by IBM Disk and Advanced BASIC.

*See: COM, ON COM.*

**operand**

An operand is a data item associated with an operator. A given operand is associated with one binary operator, and may, in addition, be associated with a unary operator.

**operator**

An operator is a symbol that represents an arithmetic, string, or logical operation. If an operator is associated with one operand, it is called a *unary* operator; if it is associated with two operands, it is called a *binary* operator. (Note that this term has nothing to do with binary numbers and the like.)

The arithmetic operators plus and minus, when used to designate the sign of a number, are unary operators. The arithmetic operators for addition, subtraction, multiplication, division, and exponentiation are binary operators.

The string operator, +, is a binary operator.

The logical operator NOT is a unary operator; the logical operators AND, OR, EQV, IMP, and XOR are binary operators.

*See: arithmetic operator, logical functions, and the individual entries.*

**OPTION  
BASE**

*Format*

OPTION BASE arithmetic-expression

The OPTION BASE statement defines the minimum value for an array subscript. It must be executed before the array is defined or used. The expression must be either 0 or 1.

*Example*

100 OPTION BASE 1

This means that any array in the program will have elements numbered from 1 to the value defined in the DIM statement, or the default. Addressing the zeroth element will cause an error.

*See: array, DIM, ERASE, subscript.*

## OPTION BASE

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft	X			
IBM	Cassette	X			
	Disk	X			
	Advanced	X			
TRS Mod III	Level I				
	Extended				
	Disk				
TRS Color	Level I				
	Extended				
	Disk				
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum	X			

**OR***Format*

argument-1 OR argument-2

OR is a logical function of two arguments. It has a value of false if both of its arguments are false, and a value of true, otherwise.

The arguments can be relations, logical variables, or anything that can be evaluated as true or false.

**P OR Q**

## Truth Table for OR

p	q	p OR q
F	F	F
F	T	T
T	F	T
T	T	T

*Example*

```
IF X > 5 OR Y < YMAX THEN GOTO 100
```

In this example, the GOTO is executed if, when the relation is evaluated, either X is greater than 5, or Y is less than YMAX, or both are true.

OR is implemented by all BASIC systems.

*See: AND, logical functions, NOT.*

## OUT

*Format*

OUT port, arithmetic-expression

The OUT statement sends a byte to the specified port. The comma is required.

The expression must be between 0 and 255, inclusive. The value output is the binary equivalent of the expression.

*See: INP.*

OUT

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft			2	
	DOS				
	Microsoft				
IBM	Cassette	X		1	
	Disk	X		1	
	Advanced	X		1	
TRS Mod III	Level I				
	Extended	X		2	
	Disk	X		2	
TRS Color	Level I				
	Extended				
	Disk				
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

*Notes*

1. The port can be between 0 and 65535.
2. The port can be between 0 and 255.

# P

## **PADDLE** *Format*

PADDLE (arithmetic-expression)

The PADDLE function has as its value the status of the designated paddle. The value returned is from 1 to 228, with 1 indicating that the paddle is all the way to the right (clockwise) and 228 indicating that it is all the way to the left (anticlockwise).

The expression must be between 1 and 7.

*See: JOYSTK, PDL, PTRIG, STICK, STRIG.*

### PADDLE

System		In	Format	Notes	Alternate Commands
APPLE	Integer				PDL
	Applesoft				PDL
	DOS				PDL
	Microsoft				PDL
IBM	Cassette				STICK
	Disk				STICK
	Advanced				STICK
TRS Mod III	Level I				
	Extended				
	Disk				
TRS Color	Level I				JOYSTK
	Extended				JOYSTK
	Disk				JOYSTK
Commodore	VIC 20				
ATARI	400/800	X			STICK
ANSI	Minimum				

## **PAINT** *Format*

PAINT (x-coordinate, y-coordinate) [,color][,border-color]

The PAINT statement fills an area with a color. Painting begins at the point designated by the x and y coordinates and continues until a line that matches the border-color is reached; any other color encountered will be painted over. If the specified point is not inside a closed figure, the entire screen will ultimately be painted.

If color is omitted, the foreground color is used; if the border-color is not specified, it defaults to the color parameter or, if none is specified, to the foreground color.

*Example*

```
100 CLS : SCREEN 1 : COLOR 1,0
120 CIRCLE (160,100), 80, 1
140 CIRCLE (160,100), 70, 1
160 CIRCLE (160,100), 60, 1
180 CIRCLE (160,100), 50, 1
200 PAINT (160,100), 2, 1
220 PAINT (160,45), 3, 1
240 PAINT (160,35), 0, 1
260 PAINT (160,25), 2, 1
```

**Output**

Four concentric circles that get painted with various colors.

*See: CIRCLE, DRAW.*

PAINT
-------

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft				
IBM	Cassette				
	Disk				
	Advanced	X		1	
TRS Mod III	Level I				
	Extended				
	Disk				
TRS Color	Level I				
	Extended	X		2	
	Disk	X		2	
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

*Notes*

1. The color and border-color must be between 0 and 3. The y-value must be between 0 and 199. In medium resolution, the x-value must be between 0 and 319; and in high resolution, between 0 and 639. (All parameters are inclusive.)

2. The color and border-color must be specified. Values can be between 0 and 8. The *x*-value must be between 0 and 255; the *y*-value, between 0 and 191. (Both parameters are inclusive.)

**PCLEAR***Format*

PCLEAR arithmetic-expression

The PCLEAR statement reserves pages for graphics memory.

The expression must be between 1 and 8, inclusive and is treated as an integer. Each page reserved is 1.5K bytes; if this statement is not used, the system default is four pages.

PCLEAR is supported only by TRS Color BASIC, Extended and Disk.

*See: PCOPY, PMODE.*

**PCLS***Format*

PCLS [color]

The PCLS statement clears the screen and sets it to the specified color. If no color is specified, the background color is used.

PCLS is supported only by TRS Color BASIC, Extended and Disk.

*See: color codes, PLOT, PRESET, PSET.*

**PCOPY***Format*

PCOPY arithmetic-expression-1 TO arithmetic-expression-2

The PCOPY statement copies one graphics page to another. Both expressions must be integers with a value between 1 and 8, inclusive. The first expression specifies the page to be copied, the second where it is to be copied to.

A page cannot be copied to one that has not been reserved by PCLEAR.

PCOPY is supported only by TRS Color BASIC, Extended and Disk.

*See: PCLEAR, PMODE.*

**PDL***Format*

PDL (paddle-number)

The PDL function has as its value the position of the specified game paddle. It returns a value between 0 and 255.

The two paddles should not be read consecutively or erroneous data may be input; so a delay such as

```
FOR X = 1 TO 10 : NEXT X
```

should be inserted between reads.

*See: BUTTON, JOYSTK, ON STRIG, PDL, STICK, STRIG.*

PDL
-----

System		In	Format	Notes	Alternate Commands
APPLE	Integer	X		1	
	Applesoft	X		2	
	DOS	X		2	
	Microsoft	X		2	
IBM	Cassette				STICK
	Disk				STICK
	Advanced				STICK
TRS Mod III	Level I				
	Extended				
	Disk				
TRS Color	Level I				JOYSTK
	Extended				JOYSTK
	Disk				JOYSTK
Commodore	VIC 20				
ATARI	400/800				PADDLE, STICK
ANSI	Minimum				

*Notes*

1. The paddle-number must be 0 or 1.
2. The paddle-number must be between 0 and 3.

**PEEK***Format*

PEEK (arithmetic-expression)

The PEEK function has as its value the contents of the memory location specified by the expression. The address and the value returned are in decimal. The address specified must exist in the computer.

*Example*

100 X = PEEK(1234)

### Output

The value of X is the contents of memory location 1234.

***See: DEF SEG, POKE.***

PEEK
------

System		In	Format	Notes	Alternate Commands
APPLE	Integer	X		1	
	Applesoft	X		1	
	DOS	X		1	
	Microsoft	X			
IBM	Cassette	X			
	Disk	X			
	Advanced	X			
TRS Mod III	Level I				
	Extended	X		1	
	Disk	X		1	
TRS Color	Level I	X			
	Extended	X			
	Disk	X			
Commodore	VIC 20	X			
ATARI	400/800	X			
ANSI	Minimum				

*Note*

1. For addresses over 32767, use the address minus 65536.

## PEN

*Format 1*

PEN ON|OFF|STOP

*Format 2*

PEN (parameter)

*Format 1*

In Cassette and Disk BASIC, this form of the PEN statement enables and disables the reading of the light pen by the PEN function (Format 2).

Before the PEN function can be used, a PEN ON must be executed. PEN OFF disables the reading of the light pen. (PEN STOP is not available.)

In Advanced BASIC, the PEN command is used to enable or disable trapping by the ON PEN statement. If PEN ON is executed, whenever any activity occurs with the light pen, a trap occurs. PEN OFF disables this trapping; if activity occurs, it is ignored and not remembered. PEN STOP disables trapping, but any activity is remembered and a trap occurs immediately after a PEN ON is executed. (Cassette I/O should not be done when trapping is activated.)

### *Format 2*

This form of PEN is a function whose value and meaning depend on the parameter supplied, as shown in the following table.

Parameter	Value of Function
0	If the pen was deactivated since the last poll, a value of $-1$ ; 0 if not
1	The $x$ -coordinate when the pen was last activated
2	The $y$ -coordinate when the pen was last activated
3	If the pen switch is currently pressed, a value of $-1$ ; 0 if not
4	The last valid $x$ -coordinate read
5	The last valid $y$ -coordinate read
6	The row where the pen was activated
7	The column where the pen was activated
8	The last valid row read
9	The last valid column read

Ranges for the foregoing values are:

Column	1 to 40 or 80, depending on WIDTH
Row	1 to 24
Y-values	0 to 199
X-values	0 to 319 (medium resolution) 0 to 639 (high resolution)

PEN is supported only by IBM (all levels).

***See: ON PEN, WIDTH.***

## **pixel**

A pixel is an addressable point on a screen. In character (text) mode, one character is made up of many pixels. In graphics mode, usually each pixel is individually addressable.

**PLAY***Format*

## PLAY string-expression

The PLAY statement generates musical tones. The note, octave, length, and tempo can be specified. There is also provision for executing substrings. The string-expression, which can be a literal or variable, is composed of the following parameters:

Parameter	Meaning	IBM Values	TRS Color Values
A to G	Name of note	A to G	A to G
+ or #	Sharp	+ or #	+ or #
—	Flat	—	—
1 to 12 or 1 to 84 or Nnumber	Note	Number can be from 1 to 84, where 1 is the C two octaves below middle C, and 84 is the B three octaves above.	Number can be from 1 to 12, where 1 = C; 2 = C#; 3 = D, etc.
Period (.)	Dotted note	Multiplies value of original note by $\frac{3}{2}$ ; two dots increase by $\frac{9}{4}$ , etc. (see note 2 below)	Increases note's value by $\frac{1}{2}$
Lvalue or L = variable;	Length of note is "1/value": 1 = whole note; 2 = half note; etc.	Value is from 1 to 64; if only 1 note is to be changed, value can follow note: L16A or A16	Value is from 1 to 255
Tvalue or T = variable;	Tempo	Number of quarter notes in a second; value can be from 32 to 255; default is 120	Value is from 1 to 255; 1 is slowest, default is 2
Pvalue or P = variable;	Rest	Values from 1 to 64; treated same as notes	Value is from 1 to 255; dots cannot be used with P
Ovalue or O = variable;	Current octave	Values from 0 to 6; an octave is from C to B; octave 3 contains middle C	Values 1 to 5; an octave is from A to G; octave 2 contains middle C; default is 2

Parameter	Meaning	IBM Values	TRS Color Values
<i>Xstring-variable;</i>	Execute the substring named by the variable. A substring can itself contain an X parameter.		Same
MF	Play music in foreground. Processing stops while the sounds play. This is the default.		Not supported
MB	Play music in background. The program continues executing while the music plays in the background. Up to 32 notes and rests can be in the buffer at one time.		Not supported
MN	Normal	A note plays for $\frac{7}{8}$ of its designated value	Not supported
ML	Legato	A note plays for its entire designated value	Not supported
MS	Staccato	A note plays for $\frac{3}{4}$ of its designated value	Not supported
Vvolume	Volume	Not supported	Range 0 to 31: 1 is softest, 0 is no sound; default is 15

*See: AUDIO, BEEP, SOUND, VARPTR\$.*

PLAY
------

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft				
IBM	Cassette				
	Disk				
	Advanced	X		1, 2	
TRS Mod III	Level I				
	Extended				
	Disk				
TRS Color	Level I				
	Extended	X		3	
	Disk	X		3	
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

*Notes*

1. To play tied notes, concatenate the expressions.
2. In conventional music notation, the first dot increases the value of a note by  $\frac{1}{2}$ , the second by  $\frac{1}{4}$  of the original value, and so on. For notes with more than one dot, a disparity is introduced. For two dots, instead of  $1\frac{3}{4}$  times the original value, it is  $2\frac{1}{4}$ ; for three dots, instead of  $1\frac{7}{8}$ , it is  $3\frac{3}{8}$ .
3. The " =variable" option cannot be used.

**PLOT***Format*

PLOT x-coordinate, y-coordinate

The PLOT statement displays a point on the screen.

*See: COLOR, HLIN, HPlot, LINE, PSET, PRESET, SET, SETCOLOR, RESET, VLIN.*

PLOT
------

System		In	Format	Notes	Alternate Commands
APPLE	Integer	X		1, 2	
	Applesoft	X		1	
	DOS	X		1	
	Microsoft	X			
IBM	Cassette				PSET
	Disk				PSET
	Advanced				PSET
TRS Mod III	Level I				SET
	Extended				SET
	Disk				SET
TRS Color	Level I				SET
	Extended				SET
	Disk				SET
Commodore	VIC 20				
ATARI	400/800	X		3	
ANSI	Minimum				

*Notes*

1. In text mode the PLOT statement displays a tiny rectangular area on the screen. Each value must be between 0 and 39. In text mode or in text and graphics mode, if the y-value is over 3 and less than 48, characters are plotted in the text area of the screen.

In low-resolution graphics mode, PLOT displays a dot at the specified point. The  $y$ -value can be between 0 and 47. The color of the dot is that specified by the last COLOR statement executed. If no COLOR statement has been executed, color 0 is used. This command has no visible effect in high-resolution graphics mode.

2. Does not support high-resolution graphics.
3. In modes 3 to 8, a point is displayed at the specified location. The color is determined by the value in the color register associated with the last COLOR statement executed. (For the valid ranges, see "GRAPHICS.") To change the color register, see "SETCOLOR."

## PMODE

*Format*

PMODE [mode,] [starting-page]

The PMODE statement selects the resolution and the first memory page to be displayed. The starting-page must be an integer between 1 and 8; it specifies which 1.5K byte memory page is to be displayed. If this parameter is omitted, the page previously selected is used. At power-up, the default is page 1.

The mode can be between 0 and 4; if omitted, the current value is used. At power-up the default is mode 2. The different values of mode configure the screen as shown in the following table.

Mode	Resolution	Grid	Colors	Pages Required	Point Size (cells)	Color Set 0	Color Set 1
0	Low	128 × 96	2	1	4	Black, green	Black, buff
1	Low	128 × 96	4	2	4	Green, blue, yellow, red	Buff, orange, magenta, cyan
2	Medium	128 × 192	2	2	2	Black, green	Black, buff
3	Medium	128 × 192	4	4	2	Green, blue, yellow, red	Buff, orange, magenta, cyan
4	High	256 × 192	2	4	1	Black, green	Black, buff

PMODE is supported only by TRS Color BASIC Extended and Disk.

*See: PCLEAR, PCOPY, SCREEN.*

## POINT

*Format 1*

POINT (x-value, y-value)

*Format 2*

POINT #disk, sector, byte

*Format 1*

The POINT function tests a cell. If the cell is off, a 0 is returned; if the point is on, its color code is returned.

*Format 2*

The POINT statement positions the pointer for the specified disk to the designated sector and byte. The next disk operation will be done starting with this location. The disk parameter can be an arithmetic expression; the other two parameters can be arithmetic variables.

*See: CLS, DSKI\$, DSKO\$, NOTE, PPOINT, RESET, SCRN, SET.*

## POINT

System		In	Format	Notes	Alternate Commands
APPLE	Integer				SCRN
	Applesoft				SCRN
	DOS				SCRN
	Microsoft				SCRN
IBM	Cassette	X	1	1	
	Disk	X	1	1	
	Advanced	X	1	1	
TRS Mod III	Level I	X	1	2	
	Extended	X	1	2	
	Disk	X	1	2	
TRS Color	Level I	X	1	3	
	Extended	X	1	3	
	Disk	X	1	3	
Commodore	VIC 20				
ATARI	400/800	X	2		
ANSI	Minimum				

*Notes*

1. Only valid in graphics mode. In high-resolution graphics the  $x$ -value can be between 0 and 639; in medium resolution, between 0 and 319. For either mode, the  $y$ -value must be between 0 and 199. If a point is out of range, a  $-1$  is returned.
2. The  $x$ -value must be between 0 and 127, the  $y$ -value between 0 and 47.
3. The  $x$ -value must be between 0 and 63, the  $y$ -value between 0 and 31.

**POKE***Format*

POKE address, arithmetic-expression

The POKE statement puts the specified value into the byte at the address. This address must exist in the machine. The address is interpreted as decimal; the value put in is the binary equivalent of the expression, which must be between 0 and 255.

*See: conversion table, DEF SEG, PEEK.*

## POKE

System		In	Format	Notes	Alternate Commands
APPLE	Integer	X		1	
	Applesoft	X			
	DOS	X			
	Microsoft	X			
IBM	Cassette	X			
	Disk	X			
	Advanced	X			
TRS Mod III	Level I				
	Extended	X		1	
	Disk	X		1	
TRS Color	Level I	X			
	Extended	X			
	Disk	X			
Commodore	VIC 20	X			
ATARI	400/800	X			
ANSI	Minimum				

*Note*

1. For addresses greater than 32767, use the address minus 65536.

**POP***Format*

## POP

The POP statement is used to remove one address from the stack. In this respect it is similar to the RETURN command but no branch is performed.

After a POP has been executed, the next RETURN will go to the *second* most recently executed GOSUB. As this command manipulates what is in the domain of the operating system, it should be used with caution, if at all.

*See: GOSUB, RETURN.*

## POP

System		In	Format	Notes	Alternate Commands
APPLE	Integer	X			
	Applesoft	X			
	DOS	X			
	Microsoft	X			
IBM	Cassette				
	Disk				
	Advanced				
TRS Mod III	Level I				
	Extended				
	Disk				
TRS Color	Level I				
	Extended				
	Disk				
Commodore	VIC 20				
ATARI	400/800	X			
ANSI	Minimum				

**POS***Format*

## POS (dummy-variable)

The POS function returns a value that indicates the current horizontal position of the cursor on the screen.

*See: CSRLIN, VPOS.*

POS
-----

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft	X		1	
	DOS	X		1	
	Microsoft	X		5	
IBM	Cassette	X		5	
	Disk	X		5	
	Advanced	X		5	
TRS Mod III	Level I				
	Extended	X		2	
	Disk	X		2	
TRS Color	Level I	X		3	
	Extended	X		3	
	Disk	X		3	
Commodore	VIC 20	X		4	
ATARI	400/800				
ANSI	Minimum				

*Notes*

1. The position is relative to the left margin of the text window. The argument, although not evaluated, must be a valid expression. Positions are numbered from 0 (HTAB and TAB number from 1).
2. Returns a value between 0 and 63.
3. If the argument is  $-1$ , POS returns the current position of the print head; if 0, the cursor position.
4. Returns a value between 0 and 21.
5. The leftmost position is number 1.

**POSITION***Format 1*

POSITION file-name [,R*position*]

*Format 2*

POSITION *x*-coordinate, *y*-coordinate

*Format 1*

A format 1 POSITION statement sets the record-pointer to the start of the field specified by position. Subsequent READ and WRITE statements proceed from that point in the file. A field is defined as a sequence of characters that ends with a Carriage Return (ASCII 13).

This is a relative, not an absolute number since fields are counted from the position of the pointer when the statement is executed.

This command must be executed, preceded by a CONTROL-D, as part of a PRINT statement:

```
PRINT CHR$(4); "POSITION MYFILE, R100"
```

The position parameter can be between 0 and 32767; the default is 0.

### *Format 2*

A format 2 POSITION moves the graphics window cursor to the location specified. The cursor does not actually move until some screen I/O operation is done. For the range of valid locations by mode, see "GRAPHICS."

*See: GRAPHICS, LOCATE, SCRN.*

### POSITION

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS	X	1		
	Microsoft				
IBM	Cassette				
	Disk				
	Advanced				
TRS Mod III	Level I				
	Extended				
	Disk				
TRS Color	Level I				
	Extended				
	Disk				
Commodore	VIC 20				
ATARI	400/800	X	2		
ANSI	Minimum				

## PPOINT

### *Format*

PPOINT (x-value, y-value)

The PPOINT function tests a graphics cell. If the cell is off, a 0 is returned; if it is on, its color code is returned. This command is valid only in graphics mode.

The x-value must be between 0 and 255; the y-value, between 0 and 191. (Both ranges inclusive.)

The color codes and their meanings are:

1	Green	5	Buff
2	Yellow	6	Cyan
3	Blue	7	Magenta
4	Red	8	Orange

PPOINT is supported only by TRS Color BASIC, Extended and Disk.

*See: PLOT, POINT, PRESET, PSET, SCREEN, SCRN.*

## PR #

*Format*

PR # slot-number

The PR # statement directs output normally intended for the screen to the device associated with the specified slot number.

The slot number must be between 1 and 7. If no device is attached to the slot, the system will hang. (Use CONTROL-C to recover.) To return output to the screen, execute PR # 0.

*See: IN#, INPUT#, PRINT#.*

PR#

System		In	Format	Notes	Alternate Commands
APPLE	Integer	X			
	Applesoft	X			
	DOS	X		1	
	Microsoft				PRINT #
IBM	Cassette				PRINT #
	Disk				PRINT #
	Advanced				PRINT #
TRS Mod III	Level I				
	Extended				PRINT #
	Disk				PRINT #
TRS Color	Level I				PRINT #
	Extended				PRINT #
	Disk				PRINT #
Commodore	VIC 20				PRINT #
ATARI	400/800				
ANSI	Minimum				

*Note*

1. This command must be executed, preceded by a CONTROL-D, as part of a PRINT statement:

```
PRINT CHR$(4);" PR # 7"
```

**precedence rules**     *See: arithmetic operations.*

**PRESET**     *Format*

PRESET (x-coordinate, y-coordinate) [,color]

The PRESET statement sets the point specified by the x and y-coordinates to the color. If no color is specified, the background color is used.

*See: PLOT, POINT, PPOINT, PSET, SET, RESET.*

PRESET
--------

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft				
IBM	Cassette	X		1	
	Disk	X		1	
	Advanced	X		1	
TRS Mod III	Level I				RESET
	Extended				RESET
	Disk				RESET
TRS Color	Level I				RESET
	Extended	X		2	RESET
	Disk	X		2	RESET
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

*Notes*

1. The  $y$ -values must be between 0 and 199. The  $x$ -value must be between 0 and 319 for medium resolution and between 0 and 639 for high resolution. (All parameters are inclusive.) In high resolution, if color 2 is specified it is taken as color 0, and color 3 is taken as color 1.
2. The  $x$ -values must be between 0 and 255;  $y$ -values, between 0 and 191, inclusive. A color cannot be specified.

**PRINT***Format*

PRINT [expression] [;|,] ...

The PRINT statement is used to display the specified data on the screen. A question mark (?) can be used instead of the word PRINT.

The list can consist of intermixed numeric and string expressions. String literals must be in quotes. When printed, all numeric values are followed by one space; in addition, a positive value is preceded by a space, a negative value by a minus sign. If nothing follows PRINT, a Carriage Return/Line Feed (CR/LF) sequence is output.

The screen is considered as being divided into zones. The punctuation between expressions determines where each item is printed. A semicolon or space after an expression causes the next item to be printed immediately after the preceding one. A comma after an expression causes the next item to be printed in the next zone.

If the list of expressions is terminated by a comma or semicolon, the next PRINT statement prints on the same line, in either the next zone or adjacent to the last data output, depending on which separator was used. Otherwise, a CR/LF is output.

If the data to be output exceeds the defined width, printing continues on the next line.

*See: HTAB, LPRINT, SPACES, SPC, TAB, VTAB, WIDTH.*

PRINT
-------

System		In	Format	Notes	Alternate Commands
APPLE	Integer	X		1, 2	
	Applesoft	X		3, 4	
	DOS	X		3, 4	
	Microsoft	X		5, 9	
IBM	Cassette	X		5, 6	
	Disk	X		5, 6	
	Advanced	X		5, 6	
TRS Mod III	Level I	X		8	
	Extended	X		8	
	Disk	X		8	
TRS Color	Level I	X		2	
	Extended	X		2	
	Disk	X		2	
Commodore	VIC 20	X		7	
ATARI	400/800	X		10	
ANSI	Minimum	X			

*Notes*

1. A question mark cannot be used instead of the word PRINT.
2. There must be a separator between elements; spaces are not permitted.
3. The zones are 16, 16, and 8 positions. The second zone is not available if there is a character in position 16. The last zone is not available if there is anything in positions 24 to 32.
4. If a semicolon is used, no spaces will be inserted between numeric values.
5. Zones are 14 spaces each.
6. If exactly 80 characters are printed, an extra Line Feed is generated, irrespective of the width setting, unless the width is "infinite."
7. There are two zones of 20 positions each.
8. There are four positions of 16 positions each.
9. If exactly 40 characters are printed, an extra Line Feed is generated.
10. A comma positions to the next tab location; the defaults are positions 7, 15, 23, 31, and 39.

**PRINT  
USING***Format*

PRINT USING format-string; {expression} ...

The PRINT USING statement specifies the formatting that is to be performed when the items are being output. The format string can be a literal

or in a variable. If it is a literal, it must be in quotes. The semicolon after the format-string is required.

The contents of the format-string determine the editing that is done on the output before it is printed. Following is a list of the valid characters and their meaning.

**Number Sign (#)**

Specifies a digit position. If the number of digits printed is less than the number of #’s, places to the right of the decimal point are filled with zeros; places to the left of the most significant digit are filled with blanks. If the number to be output is larger than the field, a percent sign (%) is printed in front of the number.

**Period (.)**

Specifies the location of the decimal point in a number.

**Comma (,)**

When placed in any position to the left of the decimal point, a comma is printed to the left of every third digit in the number.

**Two Asterisks (\*\*)**

When put at the beginning of a field, all unused positions to the left of the decimal point are filled with asterisks. This adds two positions to the size of the field.

**Dollar Sign (\$)**

Specifies that a dollar sign is to be printed in the position it occupies.

**Two Dollar Signs (\$\$)**

Specifies a floating dollar sign. The dollar sign will “float” to the right until it is immediately to the left of the most significant digit, or the decimal point, whichever comes first.

**Two asterisks and a dollar sign (\*\*\$)**

Specifies that the dollar sign floats as described above, and, in addition, any vacant positions to the left of the dollar sign are filled with asterisks.

**[[[] or ↑↑↑ or ^^^^**

Specifies that the number is to be printed in exponential format.

**Plus Sign (+)**

Specifies that a positive number prints with a plus sign and a negative number with a minus sign in the position indicated. This symbol can be to the extreme right or extreme left of the field.

**Minus Sign (—)**

Specifies that a positive number prints with a space and a negative number with a minus sign in the position indicated. This symbol can be to the extreme right or extreme left of the field.

*Characters Affecting Strings***Exclamation Point (!)**

Causes only the first character of the associated string to be printed.

**Two Percent Signs or Backslashes (% % or \ \)**

Specifies a string of spaces whose length is equal to 2 plus the number of spaces between the two signs.

Any Character before or after the String Proper Is Printed.

For example, specifying “ABC#.#.#XYZ” will cause ABC to be printed, a four-digit number with decimal point, and then XYZ.

## PRINT USING

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft	X		1, 5	
IBM	Cassette	X		2, 5	
	Disk	X		2, 5	
	Advanced	X		2, 5	
	Level I				
TRS Mod III	Extended	X		4	
	Disk	X		4	
	Level I				
TRS Color	Extended	X		2, 3	
	Disk	X		2, 3	
	Level I				
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

*Notes*

1. Either ^^^^ or ↑↑↑↑ can be used for exponential form.
2. ^^^^ is used for exponential form.
3. [ ] [ ] is used for exponential form.

4. Either [ ] or ↑↑↑↑ can be used for exponential form.
5. A variable-length field can be specified by using the ampersand (&). An underscore (\_) specifies that the next character is a literal. \ \ is used for spaces.

**PRINT@***Format*

PRINT@ position, expression

The PRINT@ statement prints the expression at the specified screen location. The comma must be included.

PRINT@

System	In	Format	Notes	Alternate Commands
APPLE	Integer			
	Applesoft			
	DOS			
	Microsoft			
IBM	Cassette			
	Disk			
	Advanced			
TRS Mod III	Level I	X	1, 2, 3	
	Extended	X	2, 3, 4	
	Disk	X	2, 3, 4	
TRS Color	Level I	X	5	
	Extended	X	5	
	Disk	X	5	
Commodore	VIC 20			
ATARI	400/800			
ANSI	Minimum			

*Notes*

1. PRINT AT must be used.
2. Positions are calculated as 0 to 63 for the first line, 64 to 127 for the second line, and so on, up to 960 to 1023 for the sixteenth line.
3. If anything is printed on the bottom line, an automatic Line Feed is executed unless a trailing semicolon is used to suppress it.
4. There can be a list of expressions, not just one.
5. Positions are calculated as 0 to 31 for the first line, 32 to 63 for the second line, and so on, up to 480 to 511 for the sixteenth line.

**PRINT #***Format*

PRINT # file-number, [USING format-string;]  
                                   {expression;}, ...

The PRINT # statement writes data to a sequential file in the same way it would be written to the screen by a PRINT statement.

Numeric expressions should be delimited by semicolons; if commas are used, extra spaces will be inserted. String expressions *must* be delimited by semicolons.

If a string contains a comma, semicolon, Carriage Return, Line Feed, or leading blanks, it should be written with explicit quotation characters by using CHR\$(34).

The file-number must be between 1 and 15, inclusive.

*See: IN#, INPUT#, PR#.*

PRINT #
---------

System		In	Format	Notes	Alternate Commands
APPLE	Integer				PR #
	Applesoft				PR #
	DOS				PR #
	Microsoft	X		1	
IBM	Cassette	X			
	Disk	X			
	Advanced	X			
TRS Mod III	Level I				
	Extended	X		2, 3	
	Disk	X		2, 3	
TRS Color	Level I	X		3	
	Extended	X		3	
	Disk	X		3	
Commodore	VIC 20	X		4, 5	
ATARI	400/800	X		5, 6	
ANSI	Minimum				

*Notes*

1. The file must be a disk file.
2. If the total number of characters is over 248, the excess is truncated.
3. For nondisk files the file number must be -1 for cassette or -2 for the printer. The USING clause cannot be used. For a disk file the file-number is a buffer-number from 1 to 15.

4. If a file-number is 1, it is the cassette, 4 is the printer, and 8 is the disk.
5. The USING clause cannot be specified.
6. The file number can be between 1 and 7.

### program line

The following chart shows the valid line numbers and the number of characters that can be in a program line.

System		In	Format	Notes	Number of Characters
APPLE	Integer			1	239
	Applesoft			1	239
	DOS			1	239
	Microsoft			2	255
IBM	Cassette				255
	Disk				255
	Advanced				255
TRS Mod III	Level I				240
	Extended				240
	Disk				240
TRS Color	Level I				
	Extended				
	Disk				
Commodore	VIC 20				88
ATARI	400/800				120
ANSI	Minimum				72

### Note

1. Spaces are not counted in determining the length of the line.






### program statements

*See: command, function, line number, statement.*

### prompt

A prompt is a character or short message provided by BASIC or the operating system to indicate to the user that the system is ready to accept input from the keyboard. The following chart shows the prompts for the different systems.

prompt

System		BASIC	Operating System
APPLE	Integer	>	*
	Applesoft	]	*
	DOS		*
	Microsoft	OK	>
IBM	Cassette	OK	>
	Disk	OK	>
	Advanced	OK	>
TRS Mod III	Level I	>	
	Extended	> 	
	Disk	> 	TRSDOS READY
TRS Color	Level I		
	Extended		
	Disk		OK
Commodore	VIC 20	OK	
ATARI	400/800	READY	
ANSI	Minimum		

**PSET***Format 1*

PSET (x-coordinate, y-coordinate) [,color]

*Format 2*

PSET (x-coordinate, y-coordinate, color)

The PSET statement sets the point at the x and y coordinates to the specified color. If no color is specified, the foreground color is used.

*See: POINT, PPOINT, PRESET, RESET, SET.*

PSET

System		In	Format	Notes	Alternate Commands
APPLE	Integer				PLOT
	Applesoft				PLOT
	DOS				PLOT
	Microsoft				PLOT
IBM	Cassette	X	1	1	
	Disk	X	1	1	
	Advanced	X	1	1	
TRS Mod III	Level I				SET
	Extended				SET
	Disk				SET
TRS Color	Level I				SET
	Extended	X	2	2	SET
	Disk	X	2	2	SET
Commodore	VIC 20				SET
ATARI	400/800				
ANSI	Minimum				

*Notes*

1. Color can be between 0 and 3. The  $y$ -value must be between 0 and 199. In medium resolution, the  $x$ -value must be between 0 and 319; in high resolution, between 0 and 639. (All ranges inclusive.)
2. The  $x$ -values must be between 0 and 255; the  $y$ -values between 0 and 191. (Both ranges inclusive.)

**PTRIG***Format*

PTRIG (arithmetic-expression)

The PTRIG function returns a zero if the trigger button of the specified controller is pressed; otherwise, it returns a 1. The expression must be between 0 and 7.

PTRIG is supported only by ATARI BASIC.

*See: **BUTTON**, **STRIG**.*

**PUT***Format 1*

PUT #file-number [,record-number]

*Format 2*

PUT (x-coordinate-1, y-coordinate-1) —  
 (x-coordinate-2, y-coordinate-2), array-name [,action]

*Format 3*

PUT # file-number, arithmetic-expression

*Format 1*

This form of the PUT statement assigns a record number to the data in the specified buffer and moves it to the file. The file must be opened and in random access mode.

The comma is required before the record-number when one is specified. If a record-number is not specified, the next record is used. The file-number must be between 1 and 15; the record number, between 1 and 32767.

*Format 2*

This form of PUT moves the contents of the specified array into a rectangular area on the screen. The first set of coordinates specifies the upper left corner of the rectangle; the second set the lower right corner. The various forms of the action parameter and their effects are shown in the following table.

Parameter	Effect for IBM	Effect for TRS Color
PSET	Displays data on screen as they are stored.	Sets points that are set in the array
PRESET	Displays a negative image on the screen	Resets points that are set in the array
OR	Superimposes the data on the screen; existing material remains	Same action
AND	A point on the screen is set if it was previously set AND the corresponding point in the array was set	Same action
XOR	A point on the screen is inverted if the corresponding point in the array is set	Not supported
NOT	Not supported	Reverses the state of each point in the array before it is plotted

*Format 3*

This form of PUT outputs a single byte, whose value is that of the expression, to the specified file. The expression can be between 0 and 255; the file number, between 1 and 7.

If the file number is 6, the screen is accessed. For screen modes 0, 1, and 2, the value is the character code; for modes 3 to 8, it is the color data. The values are the same as those in the COLOR statement.

***See: COLOR, GET, GRAPHICS, SETCOLOR.***

PUT
-----

System	In	Format	Notes	Alternate Commands
APPLE	Integer			
	Applesoft			
	DOS			
	Microsoft	X	1	
IBM	Cassette			
	Disk	X	1	
	Advanced	X	1, 2	1, 2
TRS Mod III	Level I			
	Extended			
	Disk	X	1	3
TRS Color	Level I			
	Extended	X	2	4
	Disk	X		3
Commodore	VIC 20			
ATARI	400/800	X	3	
ANSI	Minimum			

*Notes*

1. The ending coordinates cannot be specified.
2. If OR, AND, or XOR are specified, the resulting color is a function of the existing color and the color of the corresponding point in the array. The following tables give the resulting color for all combinations of screen color and array color.

Screen Color	AND				OR				XOR			
	Array Color				Array Color				Array Color			
	0	1	2	3	0	1	2	3	0	1	2	3
0	0	0	0	0	0	1	2	3	0	1	2	3
1	0	1	0	1	1	1	2	3	1	0	3	2
2	0	0	2	2	2	3	2	3	2	3	0	1
3	0	1	2	3	3	3	3	3	3	2	1	0

3. The number sign ( # ) is not allowed.
4. If in GET the G option was specified, an action must be specified. If PMODE is 0, 1, or 3, an action cannot be specified.

# R

## **RAD**

*Format*

**RAD**

The **RAD** command causes the trigonometric functions to interpret their argument in radians. This stays in effect until a **DEG** command is executed. On power-up, **RAD** is the default.

**RAD** is supported only by ATARI BASIC.

*See: ATN, COS, DEG, SIN, TAN, trigonometric functions.*

## **RAM**

*See: random access memory.*

## **RANDOM**

*Format 1*

**RANDOM**

*Format 2*

**RANDOMIZE** [value]

The **RANDOM** statement reseeds the random number generator. If executed at the start of the program, an unpredictable sequence of pseudo-random numbers will be obtained. Otherwise, the same sequence will be obtained each time the program is run from the initial state.

*See: RND.*

## RANDOM

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft	X	2	1	
IBM	Cassette	X	2	1	
	Disk	X	2	1	
	Advanced	X	2	1	
TRS Mod III	Level I				
	Extended	X	1		
	Disk	X	1		
TRS Color	Level I				
	Extended				
	Disk				
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum	X	2		

*Note*

1. If no value is included, the program halts and asks for one. The value must be between  $-32768$  and  $+32767$ .

**random access memory**

A random access memory (RAM) is a read/write memory with the property that the access time is the same for every element, and does not depend on the last element referenced.

The main memory of computers is a random access memory. Note that the term does not connote any specific technology for implementation of the memory.

**READ***Format 1*

READ {variable}...

*Format 2*

READ file-name [,Rrecord] [,Bbyte]

*Format 1*

This form of the READ statement obtains a value from a DATA statement and sets a specified variable equal to this value.

READ statements access the DATA elements in order, from the lowest-numbered DATA statement in the program to the highest-numbered. Within a DATA statement, elements are accessed from left to right. That is, the first READ executed in a program accesses the first element of the DATA statement with the lowest line number; the next READ accesses the second element of this DATA statement or, if there is no second element, the first element of the DATA statement with the next higher line number; and so on. This process continues until all the variables in the statement have a value. If the end of all the DATA elements is reached, an attempt to READ another value will cause an error.

The data read in must match the variable with respect to type (string versus numeric); however, string and numeric data can be intermixed in the DATA statement.

### *Example*

```
100 DATA "    ABC,DE:F    ",    ABC"DEF    , ABC,DEF
120 READ A$, B$, C$
140 PRINT "#";A$;"#"
160 PRINT "#";B$;"#"
180 PRINT "#";C$;"#"
```

### **Output**

```
#    ABC,DE:F    #
# ABC"DEF #
# ABC #
```

### *Format 2*

This form of READ causes subsequent INPUT and GET statements to obtain their data from the specified file instead of the keyboard. If a byte value is specified, reading begins at that byte within the specified record. Characters are read from the file one field at a time. A field can contain from 1 to 32767 characters, and is defined as a sequence of characters ending with a Carriage Return (ASCII 13).

For a sequential text file the record parameter cannot be specified. For a random access file, if a record is specified, reading begins with that record. In either type of file reading begins with the first byte in the specified record unless the byte parameter is specified; in this case, reading begins at that byte.

Both the record and byte parameter can be between 0 and 32767; the default for each is 0.

This command must be executed, preceded by a CONTROL-D, as part of a PRINT statement:

```
PRINT CHR$(4); "READ MYFILE, R100"
```

This form of READ is canceled by printing any DOS command; a PRINT CHR\$(4) is sufficient.

*See: DATA, INPUT, INPUT#, RESTORE.*

READ
------

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft	X	1		
	DOS	X	2		
	Microsoft	X	1		
IBM	Cassette	X	1		
	Disk	X	1		
	Advanced	X	1		
TRS Mod III	Level I	X	1		
	Extended	X	1		
	Disk	X	1		
TRS Color	Level I	X	1		
	Extended	X	1		
	Disk	X	1		
Commodore	VIC 20	X	1		
ATARI	400/800	X	1		
ANSI	Minimum	X	1		

### read-only memory

A read-only memory (ROM) is a random access memory that can be read from, but not written to. ROM is commonly used to store the BASIC interpreter or the like. It is sometimes referred to as "firmware."

Information in ROMs is generally put in by the manufacturer using a process known as "blasting"; once blasted, the contents are permanent.

Certain ROMs can be written to by special devices; these are called programmable read-only memories or PROMs.

### real number

A real number is a single- or double-precision number. It is also called a floating-point number. The difference between single and double precision is not in the magnitude of the number that can be represented, but in the number of digits in the representation of the number in exponential notation. (In exponential notation there is at most one digit to the left of the

decimal point and the number is followed by an exponent that determines its magnitude.)

A single-precision number's exponent is represented by "E+ee", a double-precision number by "D+ee", where "ee" is a two-digit number representing the exponent.

### Example

The number 12345670000089 would be represented thus:

single precision 1.234567E + 13

double precision 1.2345670000089D + 13

The accompanying charts show the range of values for single- and double-precision numbers.

*See: integer.*

SINGLE PRECISION					real numbers
System		Notes	Digits Stored	Digits Printed	Range
APPLE	Integer				
	Applesoft	1	10	9	$\pm 1.0E\pm 38$
	DOS		10	9	$\pm 1.0E\pm 38$
	Microsoft		7	7	$\pm 1.70141E\pm 38$
IBM	Cassette		7	7	$\pm 1.701412E\pm 38$
	Disk		7	7	$\pm 1.701412E\pm 38$
	Advanced		7	7	$\pm 1.701412E\pm 38$
TRS Mod III	Level I			7	$\pm 1.701411E\pm 38$
	Extended			7	$\pm 1.701411E\pm 38$
	Disk			7	$\pm 1.701411E\pm 38$
TRS Color	Level I			9	$\pm 10E\pm 38$
	Extended			9	$\pm 10E\pm 38$
	Disk			9	$\pm 10E\pm 38$
Commodore	VIC 20		10	9	$\pm 1.70141184E\pm 38$ $\pm 2.93873588E\pm 39$
ATARI	400/800		10	9 or fewer	$\pm 1.0E\pm 38$
ANSI	Minimum		At least 6	At least 6	$\pm 1.0E\pm 38$

real numbers

## DOUBLE PRECISION

System		Notes	Digits Stored	Digits Printed	Range
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft		16	16	$\pm 1.70141183460469E\pm 38$
IBM	Cassette		17	16	$\pm 1.70141183460469E\pm 38$
	Disk		17	16	$\pm 1.70141183460469E\pm 38$
	Advanced		17	16	$\pm 1.70141183460469E\pm 38$
TRS Mod III	Level I				
	Extended			16	$\pm 1.7014118345544556E\pm 38$
	Disk			16	$\pm 1.7014118345544556E\pm 38$
TRS Color	Level I				
	Extended				
	Disk				
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

*Note*

1. Arithmetic operations can generate a result as large as  $1.7E+38$ ; a value less than  $3.0E-39$  is taken to be zero.

**RECALL***Format*

RECALL array-name

The RECALL statement reads in data from tape and puts them into the specified array. Since the array's name is not stored with its values, an array can be recalled with a different name from that with which it was stored.

The number of *dimensions* of the array into which data are recalled must be the same as the array from which they were stored. The total number of *elements* in the array into which the data are read must be greater than or equal to the number of elements in the array that was stored. In the case of unequal arrays, the last (or if a one-dimensional array, the only) dimension can be larger than the corresponding dimension of the array that was stored; if any other dimension is larger, the results will be scrambled.

The cassette must be connected and running before this statement is executed; the start and end of the operation are signaled by a sound. This statement can be interrupted only by RESET.

RECALL is supported only by Applesoft.

*See: array, STORE.*

## relative form

*Format*

STEP (*x*-offset, *y*-offset)

Relative form is used in IBM graphics as an alternative way of specifying a point. Normally, one specifies the absolute location of a point; in using relative form, an offset from the last point plotted is specified.

For example, assume that the last point plotted was (25,25). To specify point (50,15) using relative form one would write STEP (25,−10).

*See: last point referenced.*

## REM

*Format*

REM comments

The REM statement designates a comment-line. Anything on the line is considered a comment and has no effect on program execution. If a line has multiple statements, anything after the REM is considered a comment.

REM
-----

System		In	Format	Notes	Alternate Commands
APPLE	Integer	X			
	Applesoft	X			
	DOS	X			
	Microsoft	X		1	
IBM	Cassette	X		1	
	Disk	X		1	
	Advanced	X		1	
TRS Mod III	Level I				
	Extended	X		1	
	Disk	X		1	
TRS Color	Level I	X		1	
	Extended	X		1	
	Disk	X		1	
Commodore	VIC 20	X			
ATARI	400/800	X			
ANSI	Minimum	X			

*Note*

1. A single quote ( ' ) can be used instead of REM.

**RENAME***Format 1*

RENAME old-file-specification TO new-file-specification

*Format 2*

RENAME old-file-name, new-file-name [,Ddrive] [,Sslot] [,Vvolume]

The RENAME command gives a disk file a new name. The contents of the file are not altered. If the old file was open, it is closed; extensions must be specified for both files.

*Format 2*

This command must be executed, preceded by a CONTROL-D, as part of a PRINT statement:

PRINT CHR\$(4); "RENAME OLDFILE, NEWFILE, D1, S6"

The allowable values for parameters and their defaults are shown below. Once a value has been changed, the last value specified is the default.

Parameter	Range	Default
drive	1 or 2	Drive 1
slot	1 to 7	Slot from which DOS was booted
volume	1 to 254	Volume from which DOS was booted

*See: NAME.*

RENAME

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS	X	2		
	Microsoft				
IBM	Cassette				
	Disk				
	Advanced				
TRS Mod III	Level I				
	Extended				
	Disk				
TRS Color	Level I				
	Extended				
	Disk	X	1		
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

## RENUM

*Format*

RENUM [new-first-line-number] [,starting-line] [,increment]

The RENUM command rennumbers program lines. Renumbering begins with the specified starting-line in the program, which is given the first-line-number. The increment is then added to this number and the result is the new line number for the next line, and so on. Line numbers after GOTO, GOSUB, THEN, ON ... GOTO, ON ... GOSUB are also renumbered to reflect the new values of the lines they reference. If either the first-line-number or the increment is omitted, 10 is used; if the starting-line is not specified, the entire program is renumbered.

Note that if a program is separated into blocks by leaving large gaps between line numbers, this command will leave a program with the value of increment between *all* lines.

The RENUM command will not assign an invalid line number nor will it operate so as to change the sequence of lines in the program.

*Example*

Assume that the existing program lines are as follows:

```
100
105
110
111
115
130
147
```

After executing RENUM 100,100,20, the lines are

```
100
120
140
160
180
200
220
```

However, if the existing lines are

```
10
20
30
```

the command RENUM 15,30 would try to change line 30 to 15, thus putting it ahead of line 20. Since this would change the sequence of the program, it will not execute.

*See: AUTO, NAME.*

RENUM
-------

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft	X		1	
IBM	Cassette	X		1	
	Disk	X		1	
	Advanced	X		1	
TRS Mod III	Level I				
	Extended				NAME
	Disk				NAME
TRS Color	Level I				
	Extended	X			
	Disk	X			
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

*Note*

1. Also changes line numbers in ERL and ELSE.

## reserved word

A reserved word is one with a special meaning to BASIC. As such it cannot be used as a variable name or in any user-defined way. A reserved word is also called a keyword.

Apple, Applesoft, TRS Color, TRS Mod III, and Commodore do not allow reserved words to be embedded in user-defined names. Thus one must be careful, especially with smaller words such as AND, since "LAND" would not be a valid variable name for these systems.

IBM and Microsoft permit embedding of reserved words, but when used, reserved words must be separated by spaces.

Following is a list of reserved words and the BASICs that support them.

*See: name.*

## RESERVED WORD

	Apple	Applesoft	Microsoft	IBM	Commodore	TRS Mod III	TRS Color	ATARI	ANSI
ABS	X	X	X	X	X	X	X	X	X
ADR								X	
AND	X	X	X	X	X	X	X	X	
APPEND						X			
ASC	X	X	X	X	X	X	X	X	
AT		X							
ATN		X	X	X	X	X	X	X	X
AUDIO							X		
AUTO	X		X	X		X			
BASE			X						X
BEEP			X	X					
BLOAD				X					
BSAVE				X					
BUTTON			X						
BYE								X	
CALL	X	X	X	X					
CDBL			X	X		X			
CHAIN			X	X					
CHR\$		X	X	X	X	X	X	X	
CINT			X	X		X			
CIRCLE				X			X		
CLEAR		X	X	X		X	X		
CLOAD							X	X	
CLOADM							X		
CLOCK						X			
CLOG								X	
CLOSE			X	X	X	X	X	X	
CLR	X				X			X	
CLS				X		X	X		
CMD					X	X			
COLOR			X	X			X	X	
COLOR=	X	X							
COM				X				X	

[illegible]

	Apple	Applesoft	Microsoft	IBM	Commodore	TRS Mod III	TRS Color	ATARI	ANSI
EOF			X	X		X	X		
EQV			X	X					
ERASE				X					
ERL			X	X		X			
ERR			X	X		X			
ERROR			X	X		X			
EXEC							X		
EXP		X	X	X	X	X	X	X	X
FIELD			X	X		X			
FILES			X	X					
FIX			X	X		X	X		
FLASH		X	X						
FN		X		X	X	X	X		
FOR	X	X	X	X	X	X	X	X	X
FORMAT						X			
FRE		X	X	X	X	X		X	
FREE						X			
GET		X	X	X	X	X	X	X	
GOSUB	X	X	X	X	X	X	X	X	X
GOTO	X	X	X	X	X	X	X	X	X
GR	X	X	X						
GRAPHICS								X	
HCOLOR			X						
HCOLOR=		X							
HEX\$			X	X			X		
HGR		X	X						
HGR2		X							
HIMEM:		X							
HLIN	X	X	X						
HOME		X	X						
HPlot		X	X						
HSCRN			X						
HTAB		X	X						

	Apple	Applesoft	Microsoft	IBM	Commodore	TRS Mod III	TRS Color	ATARI	ANSI
IF	X	X	X	X	X	X	X	X	X
IMP			X	X					
IN#	X	X							
INKEY\$			X	X		X	X		
INP				X		X			
INPUT	X	X	X	X	X	X	X	X	X
INPUT#			X	X	X				
INPUT\$			X	X					
INSTR			X	X		X	X		
INT		X	X	X	X	X	X	X	X
INVERSE		X	X						
JOYSTK							X		
KEY				X					
KILL			X	X		X			
LEFT\$		X	X	X	X	X	X		
LEN	X	X	X	X	X	X	X	X	
LET	X	X	X	X	X	X	X	X	X
LINE			X	X		X	X		
LIST	X	X	X	X	X	X	X	X	
LLIST			X	X		X	X		
LOAD	X	X	X	X	X	X		X	
LOC			X	X		X			
LOCATE				X				X	
LOF			X	X		X			
LOG		X	X	X	X	X	X	X	X
LOMEM:		X							
LPOS			X	X					
LPRINT			X	X		X		X	
LSET			X	X		X			
MEM						X	X		
MERGE			X	X		X			
MIDS		X	X	X	X	X	X		
MKDS			X	X		X			



	Apple	Applesoft	Microsoft	IBM	Commodore	TRS Mod III	TRS Color	ATARI	ANSI
POP	X	X	X					X	
POS		X	X	X	X	X	X		
POSITION								X	
POSN						X			
PPOINT							X		
PR#	X	X							
PRESET				X			X		
PRINT	X	X	X	X	X	X	X	X	X
PRINT#			X	X	X				
PSET				X			X		
PTRIG								X	
PUT			X	X		X	X	X	
RAD								X	
RANDOM						X			
RANDOMIZE			X	X					X
READ		X	X	X	X	X	X	X	X
RECALL		X							
REM	X	X	X	X	X	X	X	X	X
RENAME						X			
RENUM			X	X			X		
RESET			X	X		X	X		
RESTORE		X	X	X	X	X	X	X	X
RESUME		X	X	X		X			
RETURN	X	X	X	X	X	X	X	X	X
RIGHT\$		X	X	X	X	X	X		
RND	X	X	X	X	X	X	X	X	X
ROT=		X							
RSET			X	X		X			
RUN	X	X	X	X	X	X	X	X	
SAVE	X	X	X	X	X	X		X	
SCALE=		X							
SCREEN				X			X		
SCRN			X						

	Apple	Applesoft	Microsoft	IBM	Commodore	TRS Mod III	TRS Color	ATARI	ANSI
SCRN(	X	X							
SET						X	X		
SETCOLOR								X	
SGN	X	X	X	X	X	X	X	X	X
SHLOAD		X							
SIN		X	X	X	X	X	X	X	X
SKIPF							X		
SOUND				X			X	X	
SPACE\$			X	X					
SPC			X						
SPC(		X		X	X				
SPEED=		X							
SQR		X	X	X	X	X	X	X	X
STATUS								X	
STEP	X	X	X	X	X	X	X	X	X
STICK				X				X	
STOP		X	X	X	X	X	X	X	X
STORE		X							
STR\$		X	X	X	X	X	X	X	
STRIG				X				X	
STRING\$			X	X		X	X		
SUB							X		X
SWAP			X	X					
SYS					X				
SYSTEM			X	X		X			
TAB	X		X			X	X		
TAB(		X		X	X				
TAN		X	X	X	X	X	X		X
TEXT	X	X	X						
THEN	X	X	X	X	X	X	X	X	X
TIMES\$				X		X			
TIMER							X		
TO		X	X	X	X	X	X	X	X

	Apple	Applesoft	Microsoft	IBM	Commodore	TRS Mod III	TRS Color	ATARI	ANSI
TRACE	X	X	X						
TRAP								X	
TROFF				X		X	X		
TRON				X		X	X		
USING			X	X		X	X		
USR		X	X	X		X	X	X	
VAL		X	X	X	X	X	X	X	
VARPTR			X	X		X	X		
VARPTR\$				X					
VERIFY					X	X			
VLIN	X	X	X						
VPOS			X						
VTAB	X	X	X						
WAIT		X	X	X	X				
WEND			X	X					
WHILE			X	X					
WIDTH			X	X					
WRITE			X	X					
WRITE#			X	X					
XIO								X	
XOR			X	X					
XDRAW		X							
XPLOT		X							
@						X			
&		X							

**RESET***Format 1*

RESET

*Format 2*

RESET (x-value, y-value)

*Format 1*

The RESET statement closes all open files and clears the system buffer. If all open files are disk files, this statement operates the same as a CLOSE statement with no file names after it.

*Format 2*

This form of RESET erases the point at the specified screen location.

*See: PLOT, POINT, PPOINT, PRESET, PSET, SET, SYSTEM, UNLOAD.*

RESET

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft	X	1	1	
IBM	Cassette				
	Disk	X	1		
	Advanced	X	1		
TRS Mod III	Level I				
	Extended	X	2	2	
	Disk	X	2	2	
TRS Color	Level I	X	2	3	
	Extended	X	2	3	PRESET
	Disk	X	2	3	PRESET
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

*Notes*

1. Resets directory allocation information after disks have been switched.
2. The  $x$ -values must be between 0 and 127; the  $y$ -values, between 0 and 47.
3. Text mode only. The  $x$ -values must be between 0 and 63; the  $y$ -values, between 0 and 31.

**RESTORE***Format***RESTORE**

The RESTORE statement resets the pointer used for READ statements back to the first element in the first DATA statement. The first READ executed after a RESTORE will access this element. (This is the same condition that exists when the program is first run.)

*See: DATA, READ.*

**RESTORE**

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft	X			
	DOS	X			
	Microsoft	X			
IBM	Cassette	X			
	Disk	X			
	Advanced	X			
TRS Mod III	Level I	X			
	Extended	X			
	Disk	X			
TRS Color	Level I	X			
	Extended	X			
	Disk	X			
Commodore	VIC 20	X			
ATARI	400/800	X		1	
ANSI	Minimum	X			

*Note*

1. RESTORE can be followed by a line number; in this case the pointer is set to the first item on that line.

**RESUME***Format*

RESUME [NEXT | line-number]

The RESUME statement is used in conjunction with the ON ERROR GOTO statement. It causes a return from the error-handling routine. If executed before an error occurs, the results are unpredictable.

If NEXT is specified, the return is to the statement following the one in which the error occurred. If a line-number is specified, the jump is to that line. RESUME 0, or simply RESUME, returns to the statement that caused the error.

*See: ON ERROR.*

## RESUME

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft	X		1	
	DOS	X		1	
	Microsoft	X			
IBM	Cassette	X			
	Disk	X			
	Advanced	X			
TRS Mod III	Level I				
	Extended	X			
	Disk	X			
TRS Color	Level I				
	Extended				
	Disk				
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

*Note*

1. Neither NEXT nor a line number can follow RESUME. If an error occurs in the error-handling routine, the program will hang.

**RETURN***Format***RETURN**

The RETURN statement is used to exit from a subroutine. Control is transferred to the first statement following the GOSUB that branched to the subroutine.

*See: GOSUB, POP.*

RETURN
--------

System		In	Format	Notes	Alternate Commands
APPLE	Integer	X			
	Applesoft	X			
	DOS	X			
	Microsoft	X			
IBM	Cassette	X			
	Disk	X			
	Advanced	X		1	
TRS Mod III	Level I	X			
	Extended	X			
	Disk	X			
TRS Color	Level I	X			
	Extended	X			
	Disk	X			
Commodore	VIC 20	X			
ATARI	400/800	X		2	
ANSI	Minimum	X			

*Notes*

1. A line number can be specified.
2. Control returns to the following *line*, not the following statement.

**RIGHT\$***Format*

**RIGHT\$** (string-variable, arithmetic-expression)

The RIGHT\$ function has as its value the rightmost characters of the string, as specified by the expression. If the value of the expression is greater

than the number of characters in the string, the entire string is returned; if its value is zero, the null string is returned.

*See: concatenation, INSTR, LEFT\$, MID\$, null string.*

RIGHT\$
---------

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft	X		1	
	DOS	X		1	
	Microsoft	X		2	
IBM	Cassette	X		2	
	Disk	X		2	
	Advanced	X		2	
TRS Mod III	Level I				
	Extended	X			
	Disk	X			
TRS Color	Level I	X			
	Extended	X			
	Disk	X			
Commodore	VIC 20	X			
ATARI	400/800				
ANSI	Minimum				

### Notes

1. The expression can be between 1 and 255.
2. The expression can be between 0 and 255; 0 returns a null string.

## RND

### Format

RND (arithmetic-expression)

The RND function has as its value a pseudo-random number.

*See: RANDOM.*

RND

System		In	Format	Notes	Alternate Commands
APPLE	Integer	X		1	
	Applesoft	X		3	
	DOS	X		3	
	Microsoft	X		2, 4	
IBM	Cassette	X		2, 4	
	Disk	X		2, 4	
	Advanced	X		2, 4	
TRS Mod III	Level I	X		6	
	Extended	X		6	
	Disk	X		6	
TRS Color	Level I	X		7	
	Extended	X		7	
	Disk	X		7	
Commodore	VIC 20	X		5, 8	
ATARI	400/800	X		9	
ANSI	Minimum	X			

### Notes

1. If the expression is positive, the random number is between zero and one less than the expression, inclusive. The random number generator is seeded in such a way that the series cannot be reproduced.
2. If the expression is greater than zero, the number returned is a single-precision number greater than zero and less than 1; a new value is returned each time the function is called.

If the expression is less than zero, it seeds the random number generator. For a given seed value the same sequence of random numbers is generated when positive arguments are subsequently used. Each seed gives a difference sequence of numbers.

If the expression is equal to zero, the number returned is the same as the last number generated. This is useful when debugging.

3. Same rules as note 2, except that for an expression greater than zero, the number returned can be equal to zero.
4. The expression is not required; if omitted, a positive value is assumed.
5. To seed the random number generator randomly, use RND ( -TI).

6. If the expression is zero, a value greater than zero and less than 1 is returned. For an expression greater than zero, RND uses the integer part of the expression and returns a value between 1 and the expression, inclusive.
7. The expression must be greater than 1; the value returned is between 1 and the expression, inclusive.
8. Same as note 2, but if the expression is zero the random number generator is seeded from a free-running clock.
9. The number returned is less than 1 and greater than or equal to zero. The expression is a dummy variable.

**ROM**

*See: read-only memory.*

**ROT**

*Format*

ROT = arithmetic-expression

The ROT statement sets the angular rotation for a shape that will be drawn by DRAW or XDRAW.

The expression determines the amount of the rotation. It must be between 0 and 255; its value is taken modulo 64. Zero is defined as no rotation; 16 is 90-degree clockwise rotation; 32 is 180-degree rotation; and 48 is 270-degree clockwise rotation.

If SCALE is 1, only four values are recognized: 0, 16, 32, and 48. If SCALE is 2, eight values are recognized: 0, 8, 16, 24, 32, 40, 48, and 56; and so on. If a specified value is not recognized, the shape will be drawn with the next smaller recognized rotation.

ROT is parsed as a reserved word only if the next nonspace character is an equal sign (=). This means that

```
100 ROT = 10
120 PRINT ROT
```

will result in 0 being printed.

ROT is supported only by Applesoft and Apple DOS.

*See: DRAW, SCALE, shape, SHLOAD, XDRAW.*

**rounding**

Rounding is the process of deleting one or more low-order digits from a number and possibly altering the value of the remainder of the number as a function of the part deleted.

The most common form of rounding is the “5/4” system. To illustrate this system, assume that one has a number with  $K + N$  digits and one wishes to round to  $K$  digits. A value of 5 is added to the  $(K + 1)$ st digit and the resulting first  $K$  digits are taken as the rounded value. It can be seen that the value of the  $K$ th digit will be increased by 1 if the  $(K + 1)$ st digit is 5 or more.

*See: truncation.*

## RSET

*Format*

RSET field-name = character-string

The RSET statement moves the character-string into the specified field-variable, right justified. If the character-string is smaller than the field, space fill occurs on the left.

The character-string can be a literal or in a string variable. Numeric values must be converted to strings before they are RSET. Note that a field-variable is not the same as a string variable, and, that if a field-variable is used to the left of an assignment statement, it is no longer recognized as a field-variable.

*See: FIELD, LSET.*

### RSET

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft	X		1	
IBM	Cassette				
	Disk	X		1	
	Advanced	X		1	
TRS Mod III	Level I				
	Extended				
	Disk	X			
TRS Color	Level I				
	Extended				
	Disk	X			
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

*Note*

1. RSET can be used with a nonfielded string to right-justify it. For example:

```

100 A$ = SPACES(9)
120 N$ = "ABC"
140 RSET A$ = N$
160 PRINT " #";A$;" #"
```

Output:   #                   ABC#

**RUN***Format 1*

RUN [line-number]

*Format 2*

RUN program-name | file-specification [,R]

*Format 3*

RUN [file-name] [,Ddrive] [,Sslot] [,Vvolume]

The RUN command clears variables and stack space and executes a program.

*Format 1*

This form of RUN executes the program currently in memory starting at the specified line-number, which must be an actual line-number in the program. The line need not be executable, however. If it is not, execution begins with the first executable line following it. If no line-number is specified, execution begins at the first line of the program.

*Format 2*

This form of RUN loads a program from disk and runs it. If R is specified, all open files remain open for the new program; otherwise, currently open files are closed before the program is run. If no file-extension is specified, BAS is assumed.

*Format 3*

This form of RUN loads the specified file from disk and runs it. If no file-name is specified, the program in memory is run.

This command must be executed, preceded by a CONTROL-D, as part of a PRINT statement:

PRINT CHR\$(4); "RUN MYFILE, D2, S5"

The allowable values for parameters and their defaults are shown below. Once a value has been changed, the last value specified is the default.

Parameter	Range	Default
drive	1 or 2	Drive 1
slot	1 to 7	Slot from which DOS was booted
volume	1 to 254	Volume from which DOS was booted

RUN
-----

System		In	Format	Notes	Alternate Commands
APPLE	Integer	X	1	1	
	Applesoft	X	1	1	
	DOS	X	3		
	Microsoft	X	1, 2	2	
IBM	Cassette	X	1, 2	2	
	Disk	X	1, 2	2	
	Advanced	X	1, 2	2	
TRS Mod III	Level I	X	1		
	Extended	X	1		
	Disk	X	2	3	
TRS Color	Level I	X	1	4	
	Extended	X	1	4	
	Disk	X	2	2	
Commodore	VIC 20	X	1		
ATARI	400/800	X	1, 2	4, 5	
ANSI	Minimum				

*Notes*

1. Can be used only in immediate mode.
2. Must use a file-specification, not a program-name.
3. Must use a program-name, not a file-specification.
4. A line-number cannot be specified.
5. No assumption is made about the extension; R cannot be specified.

# S

## SAVE

### *Format 1*

SAVE [file-specification] [,P|A]

### *Format 2*

SAVE file-name [,device] [,command]

### *Format 3*

SAVE file-name [,Ddrive] [,Sslot] [,Vvolume]

The SAVE command stores a program on cassette or disk.

### *Format 1*

In this form of SAVE, if an A is specified, the program is stored in ASCII format. (To MERGE a program, it must have been saved in ASCII format.)

If P is specified, the program is stored in encoded binary format. This protects the program from being subsequently listed or edited. However, there is no way to cancel this protection. If no extension is specified, BAS is assumed.

### *Format 2*

The device must be 1 or 8, where 1 is the cassette and 8 is the disk. If no device is specified, cassette is assumed.

If command is a 1, when the program is reloaded it is put into the same place it was stored from; if a 2, an end-of-tape mark is written; if a 3, actions for both 1 and 2 are taken. The file-specification can be a literal or in a string variable.

### *Format 3*

If the file being saved already exists in the same language, the original file is written over; if the file exists in a different language or type, an error occurs.

This command must be executed, preceded by a CONTROL-D, as part of a PRINT statement:

PRINT CHR\$(4); "SAVE MYFILE, D1, S6"

The allowable values for parameters and their defaults are shown below. Once a value has been changed, the last value specified is the default.

Parameter	Range	Default
drive	1 or 2	Drive 1
slot	1 to 7	Slot from which DOS was booted
volume	1 to 254	Volume from which DOS was booted

*See: CHAIN, CSAVE, CSAVEM, ENTER, LIST, LOAD, MERGE.*

SAVE
------

System		In	Format	Notes	Alternate Commands
APPLE	Integer	X	1	1	
	Applesoft	X	1	1	
	DOS	X	3		
	Microsoft	X	1		
IBM	Cassette	X	1		
	Disk	X	1		
	Advanced	X	1		
TRS Mod III	Level I				
	Extended				
	Disk	X	1	2	
TRS Color	Level I				
	Extended				
	Disk	X	1	2	
Commodore	VIC 20	X	2		
ATARI	400/800	X	1	3	
ANSI	Minimum				

#### Notes

1. A file-name cannot be specified. The current program is stored on tape; no verification is made to see if the cassette is hooked up and running. At the beginning and end of the operation, a tone is sounded.
2. The "P" cannot be specified.
3. The file specification must be specified and be in quotes; P and A cannot be specified. If a drive is not specified, drive 1 is used. The file is saved in tokenized form, using long interrecord gaps.

**SAVEM***Format*

SAVEM file-specification, starting-address, ending-address,  
[execution-address]

The SAVEM command stores the contents of memory, from the starting to the ending address, on disk, giving it the specified name. If no extension is specified, BIN is assumed. If an execution-address is specified, when the program is loaded, control is transferred to that address.

The file-specification must be in quotes; all addresses are treated as hexadecimal.

SAVEM is supported only by TRS Color DOS.

*See: BSAVE, CSAVE, CSAVEM, LOADM.*

**SCALE***Format*

SCALE = arithmetic-expression

The SCALE statement sets the scale size for a shape plotted by DRAW or XDRAW.

The expression must have a value between 0 and 255, inclusive. A value of 1 specifies point-for-point reproduction; 2 specifies double size, and so on, with 0 representing the largest size.

SCALE is parsed as a reserved word only if the next nonspace character is an equal sign (=). This means that

```
100 SCALE = 10
120 PRINT SCALE
```

results in 0 being printed.

SCALE is supported only by Applesoft and Apple DOS.

*See: DRAW, ROT, shape, XDRAW.*

**SCREEN***Format 1*

SCREEN mode, color-set

*Format 2*

SCREEN (row, column [,arithmetic-expression])

*Format 3*

SCREEN [mode] [,burst] [,active-page] [,visual-page]

*Format 1*

This form of the SCREEN statement sets the screen mode. If the mode is 0, the screen is set to text mode; if 1, to graphics mode.

Color-set, in conjunction with mode, determines the available colors as shown in the following table:

Color Set	Two-Color Mode	Four-Color Mode
0	Black, green	Green, yellow, blue, red
1	Black, buff	Buff, cyan, magenta, orange

Both mode and color-set must be nonnegative.

*Format 2*

This form of SCREEN is a function that has as its value the ASCII code of the character on the screen at the specified row and column. The value returned is from 0 to 255, inclusive. The row must be between 1 and 25; the column between 1 and 40 or 1 and 80, depending on the current setting of WIDTH.

If the expression is nonzero, instead of the character itself, its color attributes are returned. This is a value between 0 and 255 which contains the code for the foreground and background colors and indicates whether the character is or is not blinking. The number is interpreted as follows:

1. The foreground color is obtained by taking the value modulo 16.
2. The background color is given by subtracting the foreground color from the value taken modulo 128.
3. If the value returned is greater than 127, the character is blinking.

*Format 3*

This form of SCREEN sets the screen attributes. The screen is erased and the new mode is stored. Then the foreground color is set to white and the background and border colors to black. The mode must have a value of 0, 1, or 2. Zero represents text mode; 1, medium-resolution graphics mode; and 2, high-resolution graphics mode.

Burst enables or disables color. Its action depends on the mode; in text mode, a zero disables color, nonzero enables it. In graphics mode, a zero enables color, nonzero disables it.

The active and visual-page parameters are valid only in text mode. The active page is the page that is written to by statements that send output to the screen. The visual page is the one that is displayed on the screen; it need not be the same as the active page. Both parameters must be between 0 and 3 if the WIDTH is 80; and between 0 and 7, if the WIDTH is 40. Initially, both pages default to page 0.

If any parameter is omitted, it keeps its current value. Note that there is only one cursor, which is shared among all the pages.

*See: CSRLIN, GRAPHICS, HGR, HGR2, LOCATE, PMODE, POSITION, PPOINT, SCRN, WIDTH.*

SCREEN
--------

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft				
IBM	Cassette	X	2, 3		
	Disk	X	2, 3		
	Advanced	X	2, 3		
TRS Mod III	Level I				
	Extended				
	Disk				
TRS Color	Level I				
	Extended	X	1		
	Disk	X	1		
Commodore	VIC 20				
ATARI	400/800				GRAPHICS
ANSI	Minimum				

### screen charac- teristics

The following charts show the characteristics of the screen for text and graphics modes.

*See: GRAPHICS, HGR, HGR2, SCREEN.*

## TEXT MODE

screen characteristics

System				Lines	Characters per Line
APPLE	Integer			24	40
	Applesoft			24	40
	DOS			24	40
	Microsoft			24	40
IBM	Cassette			25	40 or 80
	Disk			25	40 or 80
	Advanced			25	40 or 80
TRS Mod III	Level I			24	32 or 64
	Extended			24	32 or 64
	Disk			24	32 or 64
TRS Color	Level I			16	32
	Extended			16	32
	Disk			16	32
Commodore	VIC 20			23	22
ATARI	400/800			24 24	40 (2 colors) 20 (5 colors)
ANSI	Minimum				

## LOW RESOLUTION GRAPHICS MODE

screen characteristics

System		In	Notes	Number Colors	X-positions	Y-positions
APPLE	Integer		1	16	40	40
	Applesoft		2	16	40	40 or 48
	DOS			16	40	40 or 48
	Microsoft			16	40	40 or 48
IBM	Cassette			4	320	200
	Disk			4	320	200
	Advanced			4	320	200
TRS Mod III	Level I			2	128	48
	Extended			2	128	48
	Disk			2	128	48
TRS Color	Level I			9	64	32
	Extended		3	4	128	96
	Disk		3	4	128	96
Commodore	VIC 20			8	22	23
ATARI	400/800		4	4 2 or 4	40 80	24 48
ANSI	Minimum					

## HIGH RESOLUTION GRAPHICS MODE

screen characteristics

System		In	Notes	Number Colors	X-positions	Y-positions
APPLE	Integer					
	Applesoft		1, 2	8	280	160 or 192
	DOS		1, 2	8	280	160 or 192
	Microsoft		2	8	280	160 or 192
IBM	Cassette			2	640	200
	Disk			2	640	200
	Advanced			2	640	200
TRS Mod III	Level I					
	Extended					
	Disk					
TRS Color	Level I					
	Extended			2	128 or 256	192
	Disk					
Commodore	VIC 20					
ATARI	400/800		3	2 or 4 1 or 2	160 320	96 192
ANSI	Minimum					

*Notes (Low Resolution)*

1. Plus four lines for text.
2. If 40 y-positions, then four lines of text are supported.
3. Also supports the 64 × 32 mode with nine colors.
4. Depends on memory; see "GRAPHICS."

*Notes (High Resolution)*

1. To support 192 Y-positions, a 24-kilobyte system is required.
2. If 160 y-positions, then four lines of text are supported.
3. Depends on memory; see "GRAPHICS."

**SCRN***Format*

SCRN (x-coordinate, y-coordinate)

The SCRN function returns a code that specifies the color of the screen at the point specified by the coordinates.

*See: color codes, HGR, HGR2, LOCATE, POINT, POSITION, PPOINT, SCREEN.*

SCRN
------

System		In	Format	Notes	Alternate Commands
APPLE	Integer	X		1, 2, 3	
	Applesoft	X		2, 3	
	DOS	X		2, 3	
	Microsoft	X		4	
IBM	Cassette				POINT
	Disk				POINT
	Advanced				POINT
TRS Mod III	Level I				POINT
	Extended				POINT
	Disk				POINT
TRS Color	Level I				POINT
	Extended				POINT
	Disk				POINT
Commodore	VIC 20				
ATARI	400/800				LOCATE
ANSI	Minimum				

### Notes

- Both coordinates must be between 0 and 39.
- In low-resolution graphics mode, either coordinate can be between 0 and 47. However, if the  $x$ -value is between 40 and 47, the color code returned is that of the point whose  $x$ -coordinate is between 0 and 7 and whose  $y$ -coordinate is the specified  $y$ -coordinate plus 16. If this resultant value is between 39 and 47 and the mode is mixed text and graphics, the value returned is of the character at that point. If this  $y$ -value is between 48 and 63, the value returned is garbage.

In text mode, SCR N returns a value between 0 and 15 which is the upper or lower 4 bits of the character at that position, depending on whether the  $y$ -coordinate is odd or even, respectively. In high-resolution graphics mode, the number that is returned is not related to the screen.

- SCR N is not parsed as a reserved word unless the next nonspace character is a left parenthesis.
- The  $x$ -coordinate must be between 0 and 39, the  $y$ -coordinate between 0 and 47.

## SET

### Format

SET ( $x$ -coordinate,  $y$ -coordinate, color)

The SET command turns on a graphics block at the location specified. All

parameters must be nonnegative; they need not be integers, since SET will truncate and use only the integer portion.

*See: PLOT, POINT, PPOINT, PRESET, PSET, RESET.*

SET
-----

System		In	Format	Notes	Alternate Commands
APPLE	Integer				PLOT
	Applesoft				PLOT
	DOS				PLOT
	Microsoft				PLOT
IBM	Cassette				PSET
	Disk				PSET
	Advanced				PSET
TRS Mod III	Level I	X		1	
	Extended	X		1	
	Disk	X		1	
TRS Color	Level I	X		2	
	Extended	X		2	
	Disk	X		2	
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

### Notes

1. The  $x$ -value must be between 0 and 127; the  $y$ -value, between 0 and 47, both inclusive. A color cannot be specified.
2. The  $x$ -value must be between 0 and 63; the  $y$ -value, between 0 and 31, both inclusive. The color must be between 0 and 8. All four dots in a block must be of the same color.

## SET-COLOR

### Format

SETCOLOR color-register, color, intensity

The SETCOLOR statement loads a color register with a specified color (hue) and intensity (luminence).

The color register must be between 0 and 4, the color between 0 and 15, and the intensity must be an even number between 0 and 14 (the higher the number, the brighter the display). Any of these parameters can be an arithmetic expression.

If SETCOLOR is not executed, the following are the defaults for the color registers.

Register	Color Code	Color	Intensity
0	2	Orange	8
1	12	Green	10
2	9	Light blue	4
3	4	Pink	6
4	0	Gray	0

SETCOLOR is supported only by ATARI BASIC

*See: COLOR, color codes, GRAPHICS.*

## SGN

*Format*

SGN (arithmetic-expression)

The SGN function has a value corresponding to the sign of the expression. If the expression is:

negative, a - 1 is returned  
 positive, a + 1 is returned  
 zero, a zero is returned

*Example*

```
100 X = -10
120 Y = SGN(X)
140 PRINT Y, SGN(0), SGN(-X)
```

**Output**

```
-1    0    1
```

SGN

System		In	Format	Notes	Alternate Commands
APPLE	Integer	X			
	Applesoft	X			
	DOS	X			
	Microsoft	X			
IBM	Cassette	X			
	Disk	X			
	Advanced	X			
TRS Mod III	Level I				
	Extended	X			
	Disk	X			
TRS Color	Level I	X			
	Extended	X			
	Disk	X			
Commodore	VIC 20	X			
ATARI	400/800	X			
ANSI	Minimum	X			

**shape**

In Apple and Microsoft BASIC, a shape is a figure produced on the screen by DRAW and XDRAW. A *shape definition* is a sequence of plotting vectors which defines a shape. When a shape definition is plotted, the shape that it specifies is generated on the screen. One or more shape definitions, together with an index, is called a *shape table*.

*See: DRAW, ROT, SCALE, SHLOAD, XDRAW.*

**SHLOAD**

*Format*

**SHLOAD**

The SHLOAD command loads a shape table from cassette tape. The table is loaded just below the location specified by HIMEM:, and then HIMEM: is set to just below the shape table.

The starting address of the table is passed to the shape drawing routines when they are executed (DRAW, XDRAW). If a second shape table is to replace the first, HIMEM: should be reset prior to loading the table to avoid wasting memory. On 16K systems, HGR clears the top 8K of memory (locations 8192 to 16383). To force SHLOAD to put the shape table below page 1, set HIMEM: to 8192 *before* executing SHLOAD. On 24K systems, either do not use HGR2, or set HIMEM: to 16384 and do not use HGR.

SHLOAD is supported only by Applesoft.

*See: DRAW, ROT, SCALE, XDRAW.*

**SIN***Format*

SIN (arithmetic-expression)

The sine function, SIN, has as its value the sine of the angle specified by the expression, which is interpreted to be in radians.

To obtain the sine of an angle, X, that is expressed in degrees, use SIN (X \* .0174532925).

If the SIN function is not implemented, the sine of an angle, expressed in radians, can be calculated by the series

$$\text{SIN } X = X - \frac{X^3}{3!} + \frac{X^5}{5!} - \frac{X^7}{7!} + \frac{(-1)^n X^{2n+1}}{(2n+1)!} \dots$$

*Example*

```
100 XR = 45 * .01745329
120 Y = SIN(XR)
140 PRINT SIN(.5235988), Y, SIN(-XR)
```

**Output**

```
.5      .7071067      -.7071067
```

(0.5235988 radian is 30 degrees.)

*See: ATN, COS, DEG, RAD, TAN, trigonometric functions.*

SIN
-----

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft	X			
	DOS	X			
	Microsoft	X			
IBM	Cassette	X			
	Disk	X			
	Advanced	X			
TRS Mod III	Level I				
	Extended	X			
	Disk	X			
TRS Color	Level I	X			
	Extended	X			
	Disk	X			
Commodore	VIC 20	X			
ATARI	400/800	X		1	
ANSI	Minimum	X			

*Note*

1. The argument can be in either degrees or radians, depending on whether DEG or RAD was used.

**single-precision number**

*See: real number.*

**SKIPF**

*Format*

SKIPF [program-name]

The SKIPF command moves the cassette tape to just beyond the specified program's position. If a program-name is not specified, the tape is moved to the end of the next program on tape.

SKIPF is supported only by TRS Color (all levels).

**SOUND**

*Format 1*

SOUND frequency, duration

*Format 2*

SOUND voice, pitch, distortion, volume

*Format 1*

The SOUND statement generates a tone of the specified frequency for the specified time.

*Format 2*

This form of SOUND causes the specified note to begin playing. The note continues to play until another sound statement with the same voice is executed, or an END is executed.

The *voice* parameter can be from 0 to 3; each voice can be manipulated independently of the others, making harmony possible.

The *distortion* parameter can be from 0 to 14; 1 is no distortion.

The *volume* parameter can be from 1 to 15, with 15 being the loudest. The total volume of all four voices should not exceed 32.

The *pitch* parameter can be from 0 to 255. The larger the number, the lower the pitch. Approximate relations between notes and values are as follows:

Octave	C	C#	D	D#	E	F	F#	G	G#	A	A#	B
One octave above middle C	60	57	53	50	47	45	42	40	37	35	33	31
Middle C	121	114	108	102	96	91	85	81	76	72	68	64
One octave below middle C	243	230	217	204	193	182	173	162	153	144	136	128

**See: AUDIO, BEEP, PLAY.**

SOUND
-------

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft				
IBM	Cassette	X	1	1	
	Disk	X	1	1	
	Advanced	X	1	1	
TRS Mod III	Level I				
	Extended				
	Disk				
TRS Color	Level I	X	1	2	
	Extended	X	1	2	
	Disk	X	1	2	
Commodore	VIC 20				
ATARI	400/800	X	2		
ANSI	Minimum				

### Notes

1. Frequency is in hertz; it must be between 37 and 32767. Duration is in clock ticks; it must be between 0 and 65525, with one tick being 0.055 second.
2. Frequency must be between 1 and 255, with 1 the lowest tone. Duration must be between 1 to 255, with each count being 0.06 second.

**SPACE\$***Format*

SPACE\$ (arithmetic-expression)

The SPACE\$ function has as its value a string of spaces, where the number of spaces is determined by the value of the expression, which must be between 0 and 255, inclusive.

*See: PRINT, SPC, STRINGS, TAB.*

SPACE\$

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft	X			
IBM	Cassette	X			
	Disk	X			
	Advanced	X			
TRS Mod III	Level I				
	Extended				
	Disk				
TRS Color	Level I				
	Extended				
	Disk				
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

**SPC***Format*

SPC (arithmetic-expression)

The SPC function can be used only in a PRINT statement. It prints the number of spaces specified by the expression after the item previously printed. If spacing goes past the right window edge, it continues on the next line. In the PRINT statement SPC must be separated from other values by semicolons or spaces, not commas.

The expression must have a value between 0 and 255, inclusive. If its value is zero, no spaces are introduced.

*See: HTAB, PRINT, SPACE\$, TAB, VTAB.*

SPC

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft	X		1	
	DOS	X		1	
	Microsoft	X		2	
IBM	Cassette	X		1, 3	
	Disk	X		1, 3	
	Advanced	X		1, 3	
TRS Mod III	Level I				
	Extended				
	Disk				
TRS Color	Level I				
	Extended				
	Disk				
Commodore	VIC 20	X		4	
ATARI	400/800				
ANSI	Minimum				

*Notes*

1. SPC is parsed as a reserved word only if the next nonspace character is a left parenthesis.
2. Permitted only in PRINT and LPRINT statements.
3. Permitted only in PRINT, PRINT#, and LPRINT statements.
4. Permitted only in PRINT and PRINT# statements.

**SPEED***Format*

SPEED = arithmetic-expression

The SPEED statement sets the speed at which characters are sent to the screen or to other I/O devices. The expression must be between 0 and 255, with 0 being the slowest speed. SPEED is parsed as a reserved word only if the next nonspace character is an equal sign (=). This means that

```
100 SPEED = 10
120 PRINT SPEED
```

results in 0 being printed.

SPEED is supported only by Applesoft and Apple DOS.

*See: DLOAD.*

**SQR***Format*

SQR (arithmetic expression)

The square root function, SQR, has as its value the positive square root of the expression, which must be nonnegative. In general, SQR executes faster than raising a number to the 0.5 power.

*Example*

```
100 X = 1234.5
120 Y = SQR(543.21)
140 PRINT SQR(X), Y, SQR(X + Y)
```

**Output**

```
35.1355    23.3069    35.4656
```

SQR
-----

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft	X			
	DOS	X			
	Microsoft	X			
IBM	Cassette	X			
	Disk	X			
	Advanced	X			
TRS Mod III	Level I				
	Extended	X			
	Disk	X			
TRS Color	Level I				
	Extended	X			
	Disk	X			
Commodore	VIC 20	X			
ATARI	400/800	X			
ANSI	Minimum	X			

**statement** A statement is an element of the BASIC language that causes an action to happen within a program.

*See: command, function.*

**STATUS** *Format 1*

STATUS

*Format 2*

STATUS #device, arithmetic-variable

*Format 1*

The STATUS function has as its value the status of the last I/O operation. The value is in the form of a byte with the following meanings for the bits:

Bit	Cassette	Serial Bus Read/Write	Tape verify or load
0		Write timeout	
1		Read timeout	
2	Short block		Short block
3	Long block		Long block
4	Unrecoverable read error		Mismatch
5	Checksum error		Checksum error
6	End of file	EOI	
7	End of tape	Device not present	End of tape

*Format 2*

The STATUS command activates a status routine for the specified device. The error message number (if any) currently associated with this device is put into the designated variable. The device parameter can be an arithmetic expression.

See the entry on error codes for the meanings of the various codes.

## STATUS

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft				
IBM	Cassette				
	Disk				
	Advanced				
TRS Mod III	Level I				
	Extended				
	Disk				
TRS Color	Level I				
	Extended				
	Disk				
Commodore	VIC 20	X	1		
ATARI	400/800	X	2		
ANSI	Minimum				

## STICK

*Format*

STICK (arithmetic-expression)

The STICK function has as its value a coordinate of a joystick. The expression must be between 0 and 3.

*See: BUTTON, JOYSTK, PADDLE, PDL, STRIG.*

## STICK

System		In	Format	Notes	Alternate Commands
APPLE	Integer				PDL
	Applesoft				PDL
	DOS				PDL
	Microsoft				PDL
IBM	Cassette	X		1	
	Disk	X		1	
	Advanced	X		1	
TRS Mod III	Level I				
	Extended				
	Disk				
TRS Color	Level I				JOYSTK
	Extended				JOYSTK
	Disk				JOYSTK
Commodore	VIC 20				
ATARI	400/800	X		2	PADDLE
ANSI	Minimum				

*Notes*

1. Stick (0) reads all four values and returns the value for 0. The other three values do not cause the function to sample; they just return values that were previously read. So stick (0) must be executed before any of the values are valid. The range of values returned depends on the joysticks used.

Value	Meaning
0	<i>x</i> -coordinate of stick A
1	<i>y</i> -coordinate of stick A
2	<i>x</i> -coordinate of stick B
3	<i>y</i> -coordinate of stick B

2. The values returned give the angle of the stick.

Value	Degrees
14	0
6	45
7	90
5	135
13	180
9	225
11	270
10	315

Zero degrees is 12 o'clock.

**STOP***Format***STOP**

The STOP statement causes the program to halt in such a way that it can subsequently be restarted at the same point. The message **BREAK IN LINE XXXX** is printed, with XXXX indicating which line the STOP statement was in, and then control returns to the command level. Open files are *not* closed.

*See: CONT, END.*

STOP

System		In	Format	Notes	Alternate Commands
APPLE	Integer				CONTROL-C
	Applesoft	X			
	DOS	X			
	Microsoft	X			
IBM	Cassette	X			
	Disk	X			
	Advanced	X			
TRS Mod III	Level I	X			
	Extended	X			
	Disk	X			
TRS Color	Level I	X			
	Extended	X			
	Disk	X			
Commodore	VIC 20	X			
ATARI	400/800	X		1	
ANSI	Minimum	X			

*Note*

- 1. Prints STOPPED AT LINE xxxx; does not turn off sounds.

**STORE**

*Format*

STORE array-name

The STORE command stores an array onto tape. The array is not stored with its name, so it can be recalled with a different name. In the command, the array-name must be written without any subscript or dimensions, since only entire arrays can be stored.

No prompt is given, and the cassette must be attached and running before this command is executed. The beginning and end of the operation are signaled by a sound.

STORE is supported only by Applesoft.

*See: array, ASC, PRINT #, RECALL.*

**STR\$**

*Format*

STR\$ (arithmetic-expression)

The STR\$ function has as its value a character string that represents the value of the expression. If the expression is positive, a leading space is inserted; if negative, a minus sign is inserted. There is no trailing space in the string. This is the inverse function of VAL.

*Example*

If X is a numeric variable with a value of 1.702345E-20, after executing A\$ = STR\$(X), the contents of A\$ will be the characters

Δ 1 . 7 0 2 3 4 5 E - 2 0

If B\$ = STR( - X), the contents of B\$ will be

- 1 . 7 0 2 3 4 5 E - 2 0

(Note: "Δ" is used to represent a space.)

*See: VAL.*

STR\$

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft	X		2	
	DOS	X		2	
	Microsoft	X			
IBM	Cassette	X			
	Disk	X			
	Advanced	X			
TRS Mod III	Level I				
	Extended	X			
	Disk	X			
TRS Color	Level I	X			
	Extended	X			
	Disk	X			
Commodore	VIC 20	X			
ATARI	400/800	X		1	
ANSI	Minimum				

*Note*

1. At most one STR\$ can be used in a comparison. That is, one cannot write

If STR\$(X) < STR\$(Y) THEN ...

2. A positive number gives no leading space.

## STRIG

*Format 1*

STRIG ON|OFF

*Format 2*

STRIG (stick-number) ON|OFF|STOP

*Format 3*

STRIG (arithmetic-expression)

*Format 1*

This form of the STRIG statement controls sampling of the joystick buttons. STRIG ON enables sampling. Once it has been executed, before each statement in the program is executed, a check is made to see if a button has been pressed. STRIG OFF disables this checking.

*Format 2*

This form of the STRIG statement enables and disables the trapping of the joystick buttons. The stick-number must be 0, 2, 4, or 6: 0 signifies button A1; 2, button B1; 4, button A2; and 6, button B2.

STRIG ON enables trapping by the ON STRIG statement. If trapping is enabled, whenever the button is pressed control transfers to the line specified in the ON STRIG statement. STRIG OFF disables this trapping; if a button is pressed, the event is not remembered. STRIG STOP disables trapping, but if a button is pressed, the event is remembered and when an ON STRIG is executed a trap takes place immediately.

*Format 3*

This form of STRIG is a function. If sampling has been enabled by a format 1 STRIG statement, it has a value whose meaning depends on the value of the expression, as shown in the following chart.

Value	Meaning If -1 Returned	Meaning If 0 Returned
0	Button A1 has been pressed since last STRIG (0)	Button A1 has not been pressed since last STRIG (0)
1	Button A1 is currently pressed	Button A1 is not currently pressed
2	Button B1 has been pressed since last STRIG (0)	Button B1 has not been pressed since last STRIG (0)
3	Button B1 is currently pressed	Button B1 is not currently pressed
4	Button A2 has been pressed since last STRIG (0)	Button A2 has not been pressed since last STRIG (0)
5	Button A2 is currently pressed	Button A2 is not currently pressed
6	Button B2 has been pressed since last STRIG (0)	Button B2 has not been pressed since last STRIG (0)
7	Button B2 is currently pressed	Button B2 is not currently pressed

*See: BUTTON, JOYSTK, ON STRIG, PDL, PTRIG, STICK.*

STRIG
-------

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft			1	PEEK
	DOS			1	PEEK
	Microsoft				BUTTON
IBM	Cassette	X	1, 2		
	Disk	X	1, 2		
	Advanced	X	1, 2, 3		
TRS Mod III	Level I				
	Extended				
	Disk				
TRS Color	Level I			2	PEEK
	Extended			2	PEEK
	Disk			2	PEEK
Commodore	VIC 20				
ATARI	400/800	X	2	3	PTRIG
ANSI	Minimum				

*Notes*

1. PEEK (−16287) is button 0, PEEK (−16286) is button 1, and PEEK (−16285) is button 2. If the value returned is over 127, the button is being pressed.
2. PEEK (65280). If the button is not being pressed, a value of 255 or 127 is returned. If the right-hand button is being pressed, a value of 126 or 254 is returned; if the left-hand button is being pressed, a value of 125 or 253 is returned.
3. The expression can be only 0 through 3. If the specified joystick button is pressed, the value is 0; otherwise, it is 1.

**string  
expression**

A string expression is:

1. A string literal, or
2. A string variable, or
3. Two string literals or variables combined by the string operator (+), or
4. A string expression, optionally in parentheses, combined with any other string expression.

*Examples*

```

"ABC"
A$
A$ + B$
A$ + "." + B$

```

*See: concatenation.*

**string  
variable**

A string variable is a variable whose name ends with a "\$" or which has as the first letter of its name a letter that has appeared in a DEFSTR statement. A string variable can hold up to 255 characters. String variables are combined by the concatenation operator (+).

In comparing strings of different lengths, if two strings are equal in all their characters, the shorter one is considered smaller. That is, "ABC" is considered shorter than (<) "ABCDE".

*See: concatenation, literal.*

**STRING\$***Format*

STRING\$ (arithmetic-expression, character)

The STRING\$ function has as its value a string composed of multiple occurrences of a character. The value of the expression, which must be between 0 and 255, determines how many instances of the character are in the string.

The character can be a string-variable, a literal in quotes, or a numeric constant. It specifies the character that is in the string. If a literal in quotes or a string-variable is specified, only the first character is used. If a numeric value is specified, it must be between 0 and 255 and is treated as an ASCII character, control character, or graphics code.

*See: SPACES\$.*

STRING\$

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft	X			
IBM	Cassette	X			
	Disk	X			
	Advanced	X			
TRS Mod III	Level I				
	Extended	X		1	
	Disk	X		1	
TRS Color	Level I				
	Extended	X			
	Disk	X			
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

*Note*

1. A string-variable cannot be specified.

**subscript**

A subscript is a value, placed in parentheses after an array name. It designates an individual element of the array. For example, in the statement  $X = A(5)$ , the number 5 is the subscript of the array, A.

The smallest subscript value can be 0 or 1, depending on the OPTION BASE setting.

*See: array, DIM, ERASE, OPTION BASE.*

**SWAP**

*Format*

SWAP variable-1, variable-2

The SWAP statement exchanges the values of the two variables. The variables must be of the same type; they can be array elements.

*Example*

```
100 A$ = "ABC"  
120 B$ = "123"  
140 X = 5  
160 Y = 10  
180 SWAP A$, B$  
200 SWAP X, Y  
220 PRINT X, Y, A$, B$
```

**Output**

10     5     123     ABC

SWAP

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft	X			
IBM	Cassette	X			
	Disk	X			
	Advanced	X			
TRS Mod III	Level I				
	Extended				
	Disk				
TRS Color	Level I				
	Extended				
	Disk				
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

**symbol**      *See: arithmetic operator.*

**syntax rule**      A syntax rule defines the way in which reserved words, symbols, and elements are arranged to form valid BASIC statements, commands, and the like. It also defines the spelling of reserved words, the number and type of characters allowed in the statement, the placement of parentheses, and so on. It may also impose restrictions on the use of the statement.

When a syntax rule is violated, a syntax error occurs.

*See: format.*

**SYS**      *Format*

SYS address

The SYS command transfers control to the machine language routine at the specified address. No parameters are passed. The address must be in the range 0 to 65535, inclusive; it can be a constant or variable. The machine language routine must end with a RTS (return from subroutine) instruction.

SYS is supported only by Commodore VIC-20 BASIC.

*See: CALL, CLOADM, DLOAD, EXEC, USR.*

**SYSTEM**      *Format*

SYSTEM

The SYSTEM command closes all files and transfers control back to the operating system.

*See: RESET, UNLOAD.*

## SYSTEM

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft	X			
IBM	Cassette				
	Disk	X			
	Advanced	X			
TRS Mod III	Level I				
	Extended	X			
	Disk	X			
TRS Color	Level I				
	Extended				
	Disk				
Commodore	VIC 20				
ATARI	400/800				DOS
ANSI	Minimum				

**TAB***Format*

TAB (arithmetic-expression)

The TAB function can appear only in a PRINT statement. It moves the cursor to the position specified by the expression. The next character will print in that position.

*See: HTAB, PRINT, PRINT@, SPACE\$, SPC, VTAB.*

TAB
-----

System		In	Format	Notes	Alternate Commands
APPLE	Integer	X		1, 5	
	Applesoft	X		5, 6	
	DOS	X		5, 6	
	Microsoft	X		2, 5	
IBM	Cassette	X		2, 5, 6	
	Disk	X		2, 5, 6	
	Advanced	X		2, 5, 6	
TRS Mod III	Level I	X		3, 4	
	Extended	X		3, 4	
	Disk	X		3, 4	
TRS Color	Level I	X		4	
	Extended	X		4	
	Disk	X		4	
Commodore	VIC 20	X		4	
ATARI	400/800				
ANSI	Minimum				

*Notes*

1. Cannot be used in a PRINT statement, must be separate.
2. If the cursor is past the position, it is moved to that position on the next line.
3. The expression is taken modulo 128.
4. The expression can be between 0 and 255, inclusive.
5. The expression can be between 1 and 255, inclusive.
6. TAB is parsed as a reserved word only if the next nonspace character is a left parenthesis.

TAN

Format

TAN (arithmetic-expression)

The tangent function, TAN, has as its value the tangent of the angle specified by the expression, which is interpreted to be in radians.

To find the tangent of an angle that is expressed in degrees, use TAN (X \* .0174532925).

If the TAN function is not implemented, the tangent of an angle can be calculated by TAN (X) = SIN(X)/COS(X).

Example

```
100 XR = 45 * .0174533
120 Y = TAN(XR)
140 PRINT TAN(.5235988), Y, TAN(-XR)
```

Output

.5773503      .9999999      -.9999999

(0.5235988 radian is 30 degrees.)

See: ATN, COS, SIN, trigonometric functions.

TAN

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft	X			
	DOS	X			
	Microsoft	X			
IBM	Cassette	X			
	Disk	X			
	Advanced	X			
TRS Mod III	Level I				
	Extended	X			
	Disk	X			
TRS Color	Level I				
	Extended	X			
	Disk	X			
Commodore	VIC 20	X			
ATARI	400/800				
ANSI	Minimum	X			

**TEXT***Format***TEXT**

The TEXT statement sets the screen to text mode, that is, 40 characters per line by 24 lines.

*See: GR, HGR, HGR2, HTAB, VTAB.*

TEXT
------

System		In	Format	Notes	Alternate Commands
APPLE	Integer	X			
	Applesoft	X			
	DOS	X			
	Microsoft	X		I	
IBM	Cassette				
	Disk				
	Advanced				
TRS Mod III	Level I				
	Extended				
	Disk				
TRS Color	Level I				
	Extended				
	Disk				
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

*Note*

1. If TEXT is issued while in high-resolution graphics mode, it does not clear the screen. If issued in low-resolution graphics mode, the screen is cleared. If issued in text mode, it is equivalent to a VTAB 24.

**THEN***See: IF.***TI**

TI is a special variable supported only by the VIC-20. It starts with a value of 0 at power-up and is updated every  $\frac{1}{60}$  second.

**TIS**

TIS is a special variable supported only by the VIC-20. It contains the time in hours, minutes, and seconds. At power-up it starts at 0, but it can be set by an assignment statement. Setting TIS initializes TI.

**TIMES***Format*

TIMES

TIMES is a variable whose value is the time of day in the format hh:mm:ss, where “hh” is hours, “mm” is minutes, and “ss” is seconds.

*See: TI, TI\$, TIMER.*

TIMES

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft				
IBM	Cassette				
	Disk	X		1	
	Advanced	X		1	
TRS Mod III	Level I				
	Extended	X		2	
	Disk	X		2	
TRS Color	Level I				
	Extended				
	Disk				
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

*Notes*

1. Can be used as a statement or a variable.
2. Contains the date and time in the format

month-day-year-hour-minute-second

To set, POKE locations 16924 to 16919, with 16924 being the month and 16919 the second.

**TIMER***Format*

TIMER [= arithmetic-expression]

The TIMER function is used to read the contents of the interval timer or to

set it. The expression must be between 0 and 65535; TIMER is incremented 60 times a second; it is off during cassette and disk operations.

TIMER is supported only by TRS Color BASIC, Extended and Disk.

*See: TI, TI\$, TIME\$.*

## TRACE

*Format*

### TRACE

The TRACE statement enables the trace function. As each line is executed, its line number is displayed.

*See: DSP, NOTRACE, TRON.*

TRACE
-------

System		In	Format	Notes	Alternate Commands
APPLE	Integer	X		1	DSP
	Applesoft	X		2	
	DOS	X		2	
	Microsoft				
IBM	Cassette				TRON
	Disk				TRON
	Advanced				TRON
TRS Mod III	Level I				
	Extended				TRON
	Disk				TRON
TRS Color	Level I				TRON
	Extended				TRON
	Disk				TRON
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

*Notes*

1. A line with multiple statements is listed only once.
2. A line with multiple statements is listed once for each statement.

## transfer address

The transfer address is the address where execution of a machine language routine begins after the routine has been loaded. It is also called the "entry point."

*See: BLOAD, BSAVE, CLOAD, CLOADM, CSAVE, CSAVEM, SAVEM.*

**TRAP***Format*

TRAP line-number

The TRAP statement transfers control to the specified line number when an error is detected. The line number can be an expression. PEEK (195) has the error number, and PEEK(187)\*256+PEEK(186) the line number, where the error occurred. To disable trapping, use a line number from 32768 to 65535.

TRAP is supported only by ATARI BASIC.

*See: ERL, ERR, errors, ON ERROR.*

**trap**

A trap is a subroutine to which control is transferred when a certain activity occurs. This transfer is caused by the system; it does not take place under program control.

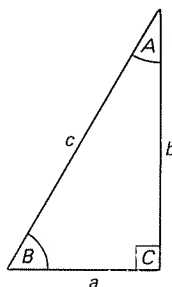
For example, executing the statement ON ERROR GOSUB 1000 enables trapping of errors. When an error subsequently occurs, the system interrupts the main program and transfers control to the subroutine at line 1000.

*See: ON COM, ON ERROR, ON KEY, ON PEN, ON STRIG.*

**trigo-  
nometric  
functions**

Certain trigonometric functions are directly supported by BASIC. Others can be derived. Following is a brief description of the more common functions.

Given a right triangle with angles  $A$ ,  $B$ , and  $C$ ; sides  $a$  and  $b$ , and hypotenuse  $c$ , certain functions can be defined:



The sine (SIN) of an angle is the ratio of the side opposite the angle to the hypotenuse:  $\text{SIN } A = a/c$ ;  $\text{SIN } B = b/c$ .

The cosine (COS) of an angle is the ratio of the side adjacent to the angle to the hypotenuse:  $\text{COS } A = b/c$ ;  $\text{COS } B = a/c$ .

It can be seen that the sine of angle  $A$  is equal to the cosine of angle  $B$ .

Since  $A + B$  must equal 90 degrees, we have

$$\sin X = \cos (90 - X)$$

$$\cos X = \sin (90 - X)$$

The tangent (TAN) of an angle is the ratio of the side opposite the angle to the side adjacent to it:  $\tan A = a/b$ ;  $\tan B = b/a$ .

The secant (SEC) of an angle is the ratio of the hypotenuse to the side adjacent to the angle:  $\sec A = c/b$ ;  $\sec B = c/a$ .

The cosecant (CSC) of an angle is the ratio of the hypotenuse to the side opposite the angle:  $\csc A = c/a$ ;  $\csc B = c/b$ .

The cotangent (COT) of an angle is the ratio of the side adjacent to the angle to the side opposite it:  $\cot A = b/a$ ;  $\cot B = a/b$ .

We also have the relations

$$\cot X = \tan (90 - X)$$

$$\sec X = \csc (90 - X)$$

These concepts can be extended to angles over 90 degrees.

Some useful relationships between angles and functions are:

$$\sin (-X) = -\sin (X)$$

$$\cos (-X) = +\cos (X)$$

$$\tan (-X) = -\tan (X)$$

$$\sin^2 X + \cos^2 X = 1$$

$$1 + \tan^2 X = \sec^2 X$$

$$1 + \cot^2 X = \csc^2 X$$

The range of the various functions are:

Function	Values from 0 to 90 Degrees
SIN	0 increasing to 1
COS	1 decreasing to 0
TAN	0 increasing to infinity
SEC	0 increasing to infinity
CSC	Infinity decreasing to 0
COT	Infinity decreasing to 0

(For values over 90 degrees, these ranges repeat themselves, possibly with a change of sign.)

*Radians*

An angle can be measured in degrees or radians. Degree measurement is fairly well known, but since a computer generates trigonometric values by series approximation, the angle must be specified in radians.

A radian is an arc of a circle whose length is equal to the circle's radius. Since the relationship between a circle's radius and circumference is  $C = 2\pi r$ , and there are 360 degrees in a circle, we can derive:

$$(1) \quad 360 = 2\pi r$$

Since a radian, by definition, is equal to a radius we can rewrite (1) as

$$(2) \quad 360 = 2\pi \text{ radians}$$

Dividing both sides by 2, we obtain

$$(3) \quad 180 = \pi \text{ radians}$$

Or

$$1 \text{ radian} = 180/\pi \text{ or } 57.29577951 \quad (\text{or } 57^\circ 17' 44.8'')$$

and

$$1 \text{ degree} = 0.0174532925 \text{ radian}$$

*See: ATN, COS, SIN, TAN.*

Following is a list of inverse functions, hyperbolic functions and inverse hyperbolic functions, and formulae for their derivation.

Function	Function Expressed in Terms of BASIC Functions (x is in radians)
Secant	$\text{SEC}(X) = 1/\text{COS}(X)$
Cosecant	$\text{CSC}(X) = 1/\text{SIN}(X)$
Cotangent	$\text{COT}(X) = 1/\text{TAN}(X)$
Inverse sine	$\text{ARCSIN}(X) = \text{ATN}(X/\text{SQR}(-X^2 + 1))$
Inverse cosine	$\text{ARCCOS}(X) = -\text{ATN}(X/\text{SQR}(-X^2 + 1)) + 1.570796327$
Inverse secant	$\text{ARCSEC}(X) = \text{ATN}(\text{SQR}(X^2 - 1)) + (\text{SGN}(X) - 1) * 1.570796327$
Inverse cosecant	$\text{ARCCSC}(X) = \text{ATN}(1/\text{SQR}(X^2 - 1)) + \text{SGN}(X) - 1 * 1.570796327$
Inverse cotangent	$\text{ARCCOT}(X) = -\text{ATN}(X) + 1.570796327$
Hyperbolic sine	$\text{SINH}(X) = (\text{EXP}(X) - \text{EXP}(-X))/2$
Hyperbolic cosine	$\text{COSH}(X) = (\text{EXP}(X) + \text{EXP}(-X))/2$
Hyperbolic tangent	$\text{TANH}(X) = -\text{EXP}(-X)/(\text{EXP}(X) + \text{EXP}(-X)) * 2 + 1$
Hyperbolic secant	$\text{SECH}(X) = 2/(\text{EXP}(X) + \text{EXP}(-X))$
Hyperbolic cosecant	$\text{CSCH}(X) = 2/(\text{EXP}(X) - \text{EXP}(-X))$
Hyperbolic cotangent	$\text{COTH}(X) = \text{EXP}(-X)/(\text{EXP}(X) - \text{EXP}(-X)) * 2 + 1$
Inverse hyperbolic sine	$\text{ARCSINH}(X) = \text{LOG}(X + \text{SQR}(X^2 + 1))$
Inverse hyperbolic cosine	$\text{ARCCOSH}(X) = \text{LOG}(X + \text{SQR}(X^2 - 1))$
Inverse hyperbolic tangent	$\text{ARCTANH}(X) = \text{LOG}((1 + X)/(1 - X))/2$
Inverse hyperbolic secant	$\text{ARCSECH}(X) = \text{LOG}((\text{SQR}(-X^2 + 1) + 1)/X)$
Inverse hyperbolic cosecant	$\text{ARCCSCH}(X) = \text{LOG}((\text{SGN}(X) * \text{SQR}(X^2 + 1) + 1)/X)$
Inverse hyperbolic cotangent	$\text{ARCCOTH}(X) = \text{LOG}((X + 1)/(X - 1))/2$

**TROFF***Format***TROFF**

The TROFF statement disables the trace function.

*See: DSP, NOTRACE, TRACE, TRON.*

TROFF

System		In	Format	Notes	Alternate Commands
APPLE	Integer				NOTRACE
	Applesoft				NOTRACE
	DOS				NOTRACE
	Microsoft				
IBM	Cassette	X			
	Disk	X			
	Advanced	X			
TRS Mod III	Level I				
	Extended	X			
	Disk	X			
TRS Color	Level I				
	Extended	X			
	Disk	X			
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

**TRON***Format***TRON**

The tron statement enables the trace function. As each program line is executed its line number is displayed. A line number is listed only once, even if there are multiple statements on the line.

*See: DSP, TRACE, TROFF.*

TRON
------

System		In	Format	Notes	Alternate Commands
APPLE	Integer				TRACE
	Applesoft				TRACE
	DOS				TRACE
	Microsoft				
IBM	Cassette	X			
	Disk	X			
	Advanced	X			
TRS Mod III	Level I				
	Extended	X			
	Disk	X			
TRS Color	Level I				
	Extended	X			
	Disk	X			
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

**truncation**

Truncation is the process of chopping off one or more digits from a number, without modifying the remainder of the number.

*See: rounding.*

**two's complement**

The two's complement of a binary number is that number obtained by adding 1 to the one's complement. (The one's complement is simply the negation of the original number; that is, all one bits are changed to 0, and all 0 bits to 1.)

Two's-complement notation is the most prevalent method of representing negative numbers. This scheme works only if the leftmost bit is reserved for the sign of the number; usually, a zero denotes a positive value, and a 1, a negative value. For example, consider the value +52 represented in a byte.

00110100	(+52)
11001011	(reverse all bits)
+        1	(add 1 to the result)
11001100	(−52)

Applying the same algorithm to a negative number results in a positive number of the same absolute value. For example, take the result of our last example:

$$\begin{array}{r}
 11001100 \quad (-52) \\
 00110011 \\
 + \quad \quad 1 \\
 \hline
 00110100 \quad (+52)
 \end{array}$$

For unsigned numbers, the concept of two's complement is not applicable.

*See: NOT.*

### **type declaration**

A variable is type declared if its name is followed by one of the characters %, !, #, or \$. A type declaration overrides a specification by a DEFINT, DEFSNG, DEFDBL, or DEFSTR statement.

Following is the meaning of each symbol.

Symbol	Meaning
%	Integer
!	Single precision
#	Double precision
\$	String

In Apple BASIC, only integers are supported, so “%” is not used. In Applesoft and ATARI BASIC, single precision is the default, so “!” is not used.

*See: DEFDBL, DEFINT, DEFSNG, DEFSTR.*

**UNLOAD***Format*

UNLOAD [drive-number]

The UNLOAD statement closes any open files on the specified drive. If no drive is specified, drive 0, or the one specified in the last DRIVE statement, is used.

*See: DRIVE, RESET, SYSTEM.*

## UNLOAD

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft				
IBM	Cassette				
	Disk				
	Advanced				
TRS Mod III	Level I				
	Extended				
	Disk	X			
TRS Color	Level I				
	Extended				
	Disk	X			
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

**UNLOCK***Format*

UNLOCK file-name [,Ddrive] [,Sslot] [,Vvolume]

The UNLOCK statement cancels the effect of a LOCK. After it is executed the file can be deleted, renamed, or altered.

This command must be executed, preceded by a CONTROL-D, as part of a PRINT statement:

```
PRINT CHR$(4); "UNLOCK MYFILE, D1, S6, V132"
```

The allowable values for parameters and their defaults are shown below. Once a value has been changed, the last value specified is the default.

Parameter	Range	Default
drive	1 or 2	Drive 1
slot	1 to 7	Slot from which DOS was booted
volume	1 to 254	Volume from which DOS was booted

UNLOCK is supported only by Apple DOS.

*See: file specification, LOCK.*

**USING** *See: PRINT USING.*

**USR** *Format 1*

USR (arithmetic-expression)

*Format 2*

USRdigit (arithmetic-expression)

*Format 3*

USR (address, [arithmetic-expression]...)

The USR statement transfers control to a user-defined machine language routine. The branch occurs when a statement such as  $X = \text{USR}(X * Y)$  is executed. The expression is passed to the routine, but it need not be used by it.

*Format 2*

The digit must be between 0 and 9, inclusive. If a digit is not specified, 0 is assumed.

*Format 3*

The address, which is interpreted in decimal, must exist in the machine. The expressions are passed as parameters to the routine and should have values between 0 and 65535. A noninteger value is rounded to an integer.

*See: CALL, CLOADM, DEF SEG, DEF USR, DLOAD, EXEC, SYS.*

USR
-----

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft	X	1	1	
	DOS	X	1		
	Microsoft	X	2		
IBM	Cassette	X	2		
	Disk	X	2		
	Advanced	X	2		
TRS Mod III	Level I				
	Extended	X	1	2, 3	
	Disk	X	2	2, 3	
TRS Color	Level I				
	Extended	X	2		
	Disk	X	2		
Commodore	VIC 20	X	1	4	
ATARI	400/800	X	3		
ANSI	Minimum				

### Notes

1. A jump to the routine must be put in locations \$0A through \$0C. The expression is evaluated, and the result put into the floating-point accumulator (\$9D to \$A3). The result, if any, is left in this accumulator.
2. The value must be between  $-32768$  and  $+32767$ ; it is passed as an integer.
3. The address of the routine must be placed in locations 16526 and 16527 (least significant byte in 16526). The expression must have an integer value.
4. The starting address of the routine must be in locations 1 and 2; the value is stored in the floating-point accumulator; when control returns to BASIC, the value is set to that of the floating-point accumulator.

# V

## VAL

### Format

VAL (string-variable)

The VAL function has as its value the numeric value of the specified string. It is the inverse function of STR\$. If the first character in the string is not numeric, a zero is returned. Any nonnumeric character after leading numeric characters is ignored. This function considers the characters +, -, E, and D as numeric. Tabs, Line Feeds, and leading spaces are ignored.

### Example

```
100 C$ = "87.65"
120 X = VAL(C$)
140 A$ = "12"
160 B$ = "34"
180 PRINT X, VAL(A$ + "." + B$)
```

### Output

87.65      13.34

(These are *numeric* values, not strings.)

**See:** STR\$.

VAL
-----

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft	X		1	
	DOS	X		1	
	Microsoft	X			
IBM	Cassette	X			
	Disk	X			
	Advanced	X			
TRS Mod III	Level I				
	Extended	X			
	Disk	X			
TRS Color	Level I	X			
	Extended	X			
	Disk	X			
Commodore	VIC 20	X			
ATARI	400/800	X			
ANSI	Minimum				

*Note*

1. If the absolute value of the number is over  $10^{37}$ , or if the number contains over 38 digits (including trailing zeros), it causes an error.

**variable**      *See: integer, name, real number, string variable.*

**VARPTR**      *Format 1*

VARPTR (variable-name)

*Format 2*

VARPTR (#file-number)

*Format 1*

This form of the VARPTR function returns the address of the first byte of the specified variable. If the variable has not been assigned a value, an error will occur.

When using an array variable-name, use ARRAY(0) to get the lowest address of the array. Also, all simple variables should be assigned before obtaining the array address.

*Format 2*

This form of VARPTR returns a value that is associated with the specified buffer.

*Example*

```
100 A$ = "ABC"
120 Y = VARPTR(A$)
140 PRINT Y
```

**Output**

3336

(Actual value will vary with implementation.)

*See: ADR, DEF SEG, VARPTR\$.*

VARPTR
--------

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft	X	1, 2	1, 2	
IBM	Cassette	X	1, 2	3, 4	
	Disk	X	1, 2	3, 4	
	Advanced	X	1, 2	3, 4	
TRS Mod III	Level I				
	Extended	X	1		
	Disk	X	1		
TRS Color	Level I	X	1		
	Extended	X	1		
	Disk	X	1		
Commodore	VIC 20				
ATARI	400/800				ADR
ANSI	Minimum				

*Notes*

1. Format 1 returns an address as an integer between  $-32768$  and  $+32767$ . To find the true address of a negative value, add 65536 to it.
2. Format 2 returns the starting address of the disk I/O buffer assigned to the file-number. If the file is a random access file, the address of the FIELD buffer is returned.
3. The value is the starting address of the file control block (FCB); this is *not* the DOS file control block.
4. The value is the offset into the current segment as defined by DEF SEG.

**VARPTR\$***Format*

VARPTR\$(variable)

The VARPTR\$ function has as its value a 3-byte string which contains both the address of the variable and a code that indicates its type. It is intended for use in DRAW and PLAY statements in programs that will be compiled. The variable must currently exist in the program.

Of the 3 bytes, the rightmost byte contains the high 8 bits of the address of the variable; the next byte contains the low 8 bits of the address of the

variable; and the leftmost byte contains a code that indicates the type of variable. These codes are:

- 2 Integer
- 3 String
- 4 Single precision
- 8 Double precision

*Example*

```
100 A$ = "ABC"
120 Z$ = VARPTR$(A$)
140 PRINT ASC(LEFT$(Z$,1)), ASC(MID$(Z$,2,1)),
      ASC(RIGHT$(Z$,1))
```

**Output**

```
3      8      13
```

$[(13 \times 256) + 8 = 3336]$ , the address of A\$.] The first byte is 3, which indicates a string variable.

In the example above, executing  $Y = \text{VARPTR}(A\$)$  results in the numeric value of 3336 for Y.

VARPTR\$ is used to indicate a variable name in the command string in a DRAW or PLAY statement:  $\text{PLAY "X"} + \text{VARPTR}(\text{B}\$)$  is equivalent to  $\text{PLAY "XB\$;"}$ .

VARPTR\$ is supported only by IBM Disk and Advanced BASIC.

*See: DRAW, PLAY, VARPTR.*

## VERIFY

*Format 1*

```
VERIFY ON|OFF
```

*Format 2*

```
VERIFY [file-name] [,device]
```

*Format 3*

```
VERIFY file-name [,Ddrive] [,Sslot] [,Vvolume]
```

*Format 1*

This form of the VERIFY command controls the verification of data written to the disk. When VERIFY ON has been executed, all data written to the disk are reread and compared against the original data. When VERIFY OFF is executed, no such verification is done.

*Format 2*

This form of VERIFY compares the specified file against the program in memory. The file-name can be a literal or in a string variable. If no file-name is specified, the first program encountered on tape is compared against the one in memory. Device 1 is the cassette; device 8 is the disk.

*Format 3*

This form of VERIFY calculates a checksum for each sector of the file and compares it against the one stored with that sector.

This command must be executed, preceded by a CONTROL-D, as part of a PRINT statement:

```
PRINT CHR$(4); "VERIFY MYFILE,D1,S6"
```

The allowable values for parameters and their defaults are shown below. Once a value has been changed, the last value specified is the default.

Parameter	Range	Default
drive	1 or 2	Drive 1
slot	1 to 7	Slot from which DOS was booted
volume	1 to 254	Volume from which DOS was booted

*See: CLOAD?.*

VERIFY
--------

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS	X	3		
	Microsoft				
IBM	Cassette				
	Disk				
	Advanced				
TRS Mod III	Level I				
	Extended				
	Disk				
TRS Color	Level I	X	1		
	Extended	X	1		
	Disk	X	1		
Commodore	VIC 20	X	2		
ATARI	400/800				
ANSI	Minimum				

**VLIN***Format*

VLIN starting-y-coordinate, ending-y-coordinate AT x-coordinate

The VLIN statement plots a vertical line from the starting to the ending *y* positions at the specified *x*-coordinate. Both *y*-values must be between 0 and 47; the *x*-value must be between 0 and 39 (both inclusive). If in text mode, or mixed text and graphics mode, then for *y*-values, between 40 and 47 characters are plotted instead of a line.

In low-resolution graphics mode the color is determined by the most recently executed COLOR statement. VLIN has no visible effect in high-resolution graphics mode.

*See: color codes, HLIN, HPLLOT, LINE, PUT.*

VLIN

System		In	Format	Notes	Alternate Commands
APPLE	Integer	X			
	Applesoft	X			
	DOS	X			
	Microsoft	X			
IBM	Cassette				LINE
	Disk				LINE
	Advanced				LINE
TRS Mod III	Level I				
	Extended				
	Disk				
TRS Color	Level I				LINE
	Extended				LINE
	Disk				LINE
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

**VPOS***Format*

VPOS (dummy-variable)

The VPOS function has the value of the line that currently contains the cursor. Values returned can range from 1 to 24, with line 1 being the top of the screen. The argument has no effect on the value the function returns.

VPOS is supported only by MicroSoft BASIC.

*See: CSRLIN, POS.*

**VTAB***Format*

VTAB arithmetic-expression

The VTAB statement moves the cursor to the vertical line designated by the expression. The horizontal position of the cursor is unchanged. The expression can be between 1 and 24, inclusive.

*See: HTAB, PRINT, PRINT@, TAB.*

VTAB

System		In	Format	Notes	Alternate Commands
APPLE	Integer	X			
	Applesoft	X		1	
	DOS	X		1	
	Microsoft	X		2, 3	
IBM	Cassette				
	Disk				
	Advanced				
TRS Mod III	Level I				
	Extended				
	Disk				
TRS Color	Level I				
	Extended				
	Disk				
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

### Notes

1. The moves are absolute with respect to the screen, not the text window.
2. If the expression is over 24, it is taken modulo 24; if it is over 255, it gives an error.
3. If a nonstandard terminal is connected, a "cursor address" sequence is sent.

# W

## WAIT

*Format*

WAIT address, mask [,pattern]

The WAIT command continually examines the bit pattern in a byte. If certain conditions are met, the program continues execution with the statement following the WAIT. If the conditions are not met, the program stays in a loop, continually testing the byte for the conditions.

The pattern must be an integer between 0 and 255; the mask must be an integer between 1 and 255. (If the mask is 0, the program will hang forever.) It is the bit pattern of these parameters that is used in the WAIT. For the bit pattern associated with different decimal values, see the entry on conversion tables.

The WAIT command operates as follows. If only the address and mask are specified, the 8 bits at the specified address are ANDed with the mask. If the result is zero, the test is repeated. When the result becomes nonzero, execution of the program continues with the statement following the WAIT. This is a way of testing whether any bit in the address that corresponds to a 1 bit in the mask is set.

### *Example*

The statement: WAIT address, 15 tests the contents of the specified address until any one of the four rightmost bits is set.

If both the mask and pattern are specified, the 8 bits in the address are XORed with the pattern. The result of this operation is then ANDed with the mask. If this result is zero, the test is repeated. When the result becomes nonzero, execution of the program continues with the instruction following the WAIT. This is a way of testing whether any bit in the address that corresponds to a one bit in the mask is in the opposite state of the corresponding bit in the pattern.

### *Example*

The statement WAIT address,3,2 tests the address to see if the rightmost bit is set or the next-to-the-rightmost bit is clear, or both conditions are true.

The logical operations are as follows:

	Case A	Case B	Case C
Address Contents	01101001	01101010	01101011
XOR with 10	00000010	00000010	00000010
Result	01101011	01101000	01101001
AND with 11	00000011	00000011	00000011
Result	00000011	00000000	00000001

In case A both bits were in the desired state, and the result was nonzero. In case B, neither bit was in the desired state and the result was zero. In case C, only the low-order bit was in the desired state and the result was non-zero.

*See: AND, mask, XOR.*

WAIT
------

System	In	Format	Notes	Alternate Commands
APPLE	Integer			
	Applesoft	X	1, 2, 3	
	DOS	X	1, 2, 3	
	Microsoft	X	1, 3	
IBM	Cassette	X	4, 5	
	Disk	X	4, 5	
	Advanced	X	4, 5	
TRS Mod III	Level I			
	Extended			
	Disk			
TRS Color	Level I			
	Extended			
	Disk			
Commodore	VIC 20	X	3	
ATARI	400/800			
ANSI	Minimum			

### Notes

1. Only RESET can interrupt a WAIT.
2. Positive and negative addresses are considered equivalent.
3. The address is a machine address.

4. The address is a port (0 to 65535).
5. CONTROL-BREAK or a system reset will interrupt the WAIT.

**WEND***Format***WEND**

The WEND statement is used to delimit the range of a WHILE statement.

*See: WHILE.*

WEND
------

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft	X			
IBM	Cassette	X			
	Disk	X			
	Advanced	X			
TRS Mod III	Level I				
	Extended				
	Disk				
TRS Color	Level I				
	Extended				
	Disk				
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

**WHILE***Format***WHILE** arithmetic-expression

The WHILE statement causes a range of instructions to be repeatedly executed until a condition becomes false.

The *range* of the WHILE consists of all the statements that follow it, down to the first WEND statement after it. The expression in the WHILE statement is evaluated. If it is not false (zero), all the statements down to the first WEND are executed. Then control returns to the WHILE statement and the expression is evaluated again. This continues until the expression becomes zero, at which time control transfers to the first statement following the WEND.

Example

```
100 X = 3
120 J = - 1
140 WHILE J <= X
160 PRINT J,
180 J = J + 1
200 WEND
```

Output

-1     0     1     2     3

See: *FOR, loop.*

WHILE

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft	X			
IBM	Cassette	X			
	Disk	X			
	Advanced	X			
TRS Mod III	Level I				
	Extended				
	Disk				
TRS Color	Level I				
	Extended				
	Disk				
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

WIDTH

Format 1

WIDTH [file-number|“device”,] size

Format 2

WIDTH [LPRINT] line-width

Format 3

WIDTH [screen-width] [,screen-height]

*Format 1*

This form of the WIDTH statement sets the output line width in terms of the number of characters on a line. After the specified number of characters is output, a Carriage Return/Line Feed (CR/LF) is automatically issued. The width must be between 0 and 255.

The width for a printer defaults to 80 when BASIC is invoked. The maximum width is 132. If a width of 255 is specified, it is interpreted as “infinite” width; that is, the CR/LF is not automatically issued, and the position of the cursor, as read by POS or LPOS, goes from 0 to 255 and then back to 0. Thus it may have no relation to the physical position of the cursor or print head.

If no file-number or device is specified, the screen width is set, in which case the size must be 40 or 80. Setting the screen width clears the screen and sets the border color to black.

If a device is specified, it must be SCRN:, LPT1:, LPT2:, LPT3:, COM1:, or COM2:. The new width value is stored without changing the current width setting. The next time the device is opened, the new width is used. (Note that LIST, “LPTn:” and LLIST perform an implicit OPEN.)

If a file-number is specified it must be between 1 and 15, inclusive. The new width is put into effect immediately. In Cassette BASIC, only the file-number associated with LPT1: can be used. In Disk and Advanced BASIC, any of the foregoing devices can be specified.

The default width for communications files is 255.

*Format 2*

This form of WIDTH sets the line width for the printer or terminal. If LPRINT is not specified, the terminal’s width is set; if it is specified, the line printer’s width is set. The maximum width is 132. If a width of 255 is specified, it is interpreted as “infinite” width; that is, the CR/LF is not automatically issued, and the position of the cursor, as read by POS or LPOS, goes from 0 to 255 and then back to 0. Thus it may have no relation to the physical position of the cursor or print head.

*Format 3*

This form of WIDTH sets the screen parameters. The width can be from 15 to 255; the height, from 1 to 24. At least one parameter must be specified. The default value for height is 24; for width it is 40 or 80, depending on the system.

**See: LPOS, POS, PRINT.**

WIDTH

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft	X	2, 3		
IBM	Cassette	X	1		
	Disk	X	1		
	Advanced	X	1		
TRS Mod III	Level I				
	Extended				
	Disk				
TRS Color	Level I				
	Extended				
	Disk				
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

## wild card

A wild card is a special character used in filenames and extensions. It is used when referencing an existing file, not when creating a new one. There are two kinds of wild cards: single-character and multiple-character.

The single-character wild card is the question mark (?). Any character in a position occupied by a question mark is considered to match the wild card. For example, if files were named MYFILE1, MYFILE2, and so on, the command `DIR MYFILE?` would list MYFILE1 through MYFILE9. The command `DIR MYFILE??` would list files MYFILE1 through MYFILE99.

The multiple-character wild card is the asterisk (\*). (In Apple DOS it is the equal sign (=).) Any character in the position occupied by the asterisk and in all positions to the right are considered to match the wild card. For example, the command `DIR ABC*` would list all files beginning with ABC and having any other characters for the rest of the name and any extension. The command `DIR *.BAS` would list all files with an extension of BAS.

(Note: In some systems the filename and extension are treated separately. To list all files, one would execute `DIR *.*`.)

## WRITE

*Format 1*

**WRITE** [expression] ...

*Format 2*

WRITE file-name [,Bbyte-address] [,Rrecord-number]

*Format 1*

This form of the WRITE statement outputs data to the screen or terminal. If no expressions are specified, a blank line is output. If one or more expressions are specified, their values are output. The expressions can be arithmetic or string, and must be separated by commas or semicolons.

When output, each value is separated from the last by a comma. Strings are delimited by quotes. After the last term is output, a Carriage Return/Line Feed (CR/LF) is output.

WRITE operates similarly to PRINT, except that WRITE inserts commas and quotes as described above, and positive numbers are not preceded by a space.

*Format 2*

This form of WRITE sends all output that would normally be displayed on the screen to the designated file instead. WRITE is canceled by PRINTing any DOS command, even just a CTL-D, or by an INPUT statement.

For a sequential file the record number cannot be specified. If the byte parameter is specified, writing begins at that byte in the record. Each record in the file must contain the number of bytes specified when the file was created.

This command must be executed, preceded by a CONTROL-D, as part of a PRINT statement:

PRINT CHR\$(4); "WRITE MYFILE, B100, R100"

Both the byte and record parameters must be between 0 and 32767, inclusive. The default for each is 0.

*See: OPEN, PRINT, PRINT#, WRITE#.*

WRITE

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS	X	2		
	Microsoft	X	1	1	
IBM	Cassette	X	1		
	Disk	X	1		
	Advanced	X	1		
TRS Mod III	Level I				
	Extended				
	Disk				
TRS Color	Level I				
	Extended				
	Disk				
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				

*Note*

1. Commas, not semicolons, must be used between elements.

## WRITE #

*Format*

WRITE # file-number, {expression} ...

The WRITE # statement writes data to a sequential file. The expressions can be string or arithmetic; they must be separated by commas and semicolons. WRITE # operates similarly to PRINT #, but in WRITE # operations, commas are put between items and strings are output with quotes around them. Also, WRITE # does not put a space in front of a positive number. After the last item is output, a Carriage Return/Line Feed (CR/LF) is output.

*See: OPEN, PRINT, PRINT #, WRITE.*

WRITE #
---------

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft	X		1	
IBM	Cassette	X			
	Disk	X			
	Advanced	X			
TRS Mod III	Level I				
	Extended				
	Disk				
TRS Color	Level I				
	Extended				
	Disk				
Commodore	VIC 20	X		1, 2	
ATARI	400/800				
ANSI	Minimum				

*Notes*

1. Commas, not semicolons, must be used between elements.
2. Only variables can be used, not expressions.

**XDRAW***Format*

XDRAW arithmetic-expression [AT x-value, y-value]

The XDRAW statement operates the same as DRAW, but the color used is the complement of the color currently at each point plotted. Two consecutive identical XDRAW statements will erase a shape without erasing the background.

XDRAW is supported only by Applesoft BASIC.

*See: DRAW, ROT, SCALE, shape, SHLOAD.*

**XIO***Format*

XIO command, #device-number, control-byte-1, control-byte-2,  
file-specification

The XIO statement is a general-purpose input-output statement.

The *command* parameter determines the action.

Command	Action	Similar to ATARI Statement
3	Open File	OPEN
5	Get Record	INPUT
7	Get Character	GET
9	Put Record	PRINT
11	Put Character	PUT
12	Close	CLOSE
13	Status Request	STATUS
17	Draw Line	DRAWTO
18	Fill	
32	Rename	
33	Delete	
35	Lock File	
36	Unlock File	
37	Point	POINT
38	Note	NOTE
254	Format	

The *device-number* selects the device; it must have a value between 1 and 7 and can be an expression. The use of the two control bytes depends on the device. The file specification is mandatory and must be in quotes.

XIO is supported only by ATARI BASIC.

## XOR

### *Format*

argument-1 XOR argument-2

The exclusive or function, XOR, is a logical function of two arguments. It has a value of false if both its arguments have the same value and a value of true, otherwise.

The arguments can be relations, logical variables, or anything that can be evaluated as true or false. This is the inverse of the equivalence function (EQV). In formal logic the exclusive or function is “P or Q but not both.”

Truth Table for XOR

p	q	p XOR q
F	F	F
F	T	T
T	F	T
T	T	F

If XOR is not implemented, it can be calculated by

DEF FNXOR (P,Q) = (P AND NOT Q) OR (NOT P AND Q)

*See: EQV, logical functions.*

XOR

System		In	Format	Notes	Alternate Commands
APPLE	Integer				
	Applesoft				
	DOS				
	Microsoft	X			
IBM	Cassette	X			
	Disk	X			
	Advanced	X			
TRS Mod III	Level I				
	Extended				
	Disk				
TRS Color	Level I				
	Extended				
	Disk				
Commodore	VIC 20				
ATARI	400/800				
ANSI	Minimum				



# THE PERSONAL COMPUTER BASIC[S] REFERENCE MANUAL

**Donald A. Sordillo**

**If you are:**

- considering purchasing a personal computer;
- writing software for commercial purposes;
- converting BASIC programs from one system to another;
- looking for a comprehensive BASIC(S) reference manual,

this book, by **Donald A. Sordillo**, is intended for you. It is a reference manual of Personal Computer BASIC(S) that covers the IBM, Apple, Atari, TRS Color, TRS Mod III, and Commodore VIC-20 computers in depth.

By concentrating on personal computers and by limiting the number of systems it covers, this book can fully document each system. The computers covered account for over 90% of personal computers, and include machines with a wide range of functionality and price, from the IBM Personal Computer to "game" machines. All of the information for all of the systems is covered.

**Among its special features, the book:**

- includes all instructions, even input/output instructions and those of the various Disk Operating Systems;
- identifies which BASIC does what;
- contains over 600 cross references and 200 charts following the BASIC keywords;
- is based on the most up-to-date information.

PRENTICE-HALL, INC., Englewood Cliffs, N.J. 07632

ISBN 0-13-658047-5